REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*
14-12-2005

2. REPORT TYPE
Final Report

3. DATES COVERED *(From – To)*
01-Dec-00 - 01-Dec-05

4. TITLE AND SUBTITLE
Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures

5a. CONTRACT NUMBER
ISTC Registration No: 1993p

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Dr. Vladimir I Gorodetsky

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
St. Petersburg Institute For Informatics & Automation of the Russian Academy of Sciences
39, 14th Liniya
St. Petersburg 199178
Russia

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
EOARD
PSC 802 BOX 14
FPO 09499-0014

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
ISTC 00-7031-1

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Global awareness (GA) entails the acquisition of data from local to global levels, appropriate fusing of the data, and presentation of that data as useful information. This data will then be fused to fully describe situations of interest such as large transportation systems and complex communication systems. This project specifically aims at developing the mathematical basis, architecture and software techniques implementing particular new technologies to support Global Awareness and comprises six main tasks. Task 1 was:
1. Autonomous Information Collection and Knowledge Discovery from Data (KDD) Techniques and Software Prototype for Knowledge-Based Data Fusion; including addendums on "Multi-agent Information Fusion for Situation Assessment and Prediction: Architecture, Case Study and Software Prototype", "Multi-agent Information Fusion for Situation Assessment and Prediction: Process Flow– and Data with Missing Value-based On-line Situation Assessment Update";" Multi-agent Technology for Airspace Deconfliction".

15. SUBJECT TERMS
EOARD, Mathematical And Computer Sciences, Computer Programming and Software

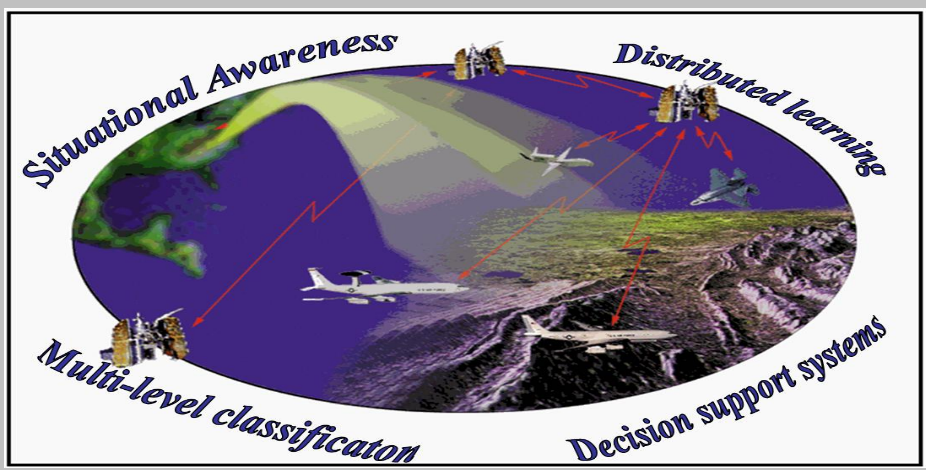
16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18, NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
UNCLAS	UNCLAS	UNCLAS	UL	40	PAUL LOSIEWICZ, Ph. D.

19b. TELEPHONE NUMBER *(Include area code)*
+44 20 7514 4474

ST. PETERSBURG INSTITUTE
FOR INFORMATICS AND AUTOMATION
(SPIIRAS)

EUROPEAN OFFICE OF AEROSPACE
RESEARCH AND DEVELOPMENT
(EOARD)

Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures



Project #1993 P

Task 1: Autonomous Information Collection, Knowledge Discovery Techniques and Software Tool Prototype for Knowledge-Based Data Fusion

Addendum 3: Multi-agent Technology for Airspace Deconfliction

Final Report

Project Manager
Head of Intelligent Systems
Laboratory of SPIIRAS

Ph.D. Professor

V.I. Gorodetsky

St. Petersburg
November 2005

ST. PETERSBURG INSTITUTE
FOR INFORMATICS AND
AUTOMATION

EUROPEAN OFFICE OF AEROSPACE
RESEARCH AND DEVELOPMENT

(EOARD)

Project # 1993P

Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures

Task 1. Autonomous Information Collection, Knowledge Discovery Techniques and Software Tool Prototype for Knowledge-Based Data Fusion

Addendum 3: Multi-agent Technology for Airspace Deconfliction

Final Report

Project Manager
Head of Intelligent System Laboratory
of the St. Petersburg Institute for Informatics and Automation

Professor Vladimir I.Gorodetsky,
Doctor of Sciences (Tech), Ph. D.

St. Petersburg
November 2005

Table of Content

Preface

This Report is final one on **Addendum 3** to the **Task#1** ("Multi-agent Information Fusion for Situation Assessment and Prediction: Architecture, Case Study and Software Prototype") of the **Project 1993P** ("Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures"). The **Addendum** entitled "*Multi-agent Technology for Airspace Deconfliction*" is scheduled for Q19–Q20 of the research on the main project.

As compared with the previous research on this Project, which was mostly of fundamental character, the research assumed by Addendum 3 is application–oriented. The main motivation behind it is to apply theoretical results of the main Project (multi-agent technology and its use within knowledge–based information fusion for situation assessment) in particular class of application that is airspace deconfliction within homeland security scenario.

Airspace deconfliction within a homeland security scenario is a typical component of many more general important problems associated with situation assessment and prediction intended to provide air traffic security and reliability. Indeed, airspace deconfliction issues the challenges peculiar to many air traffic control applications. Examples of such challenging issues are algorithms for detection and resolution of potential conflicts among objects moving in a bounded space; methods and scenarios of their realization; providing grounded decomposition and coordination of the local solutions; generic distributed architectures specifically designed for solving large scale planning, scheduling and resource allocation; real time control intended for adaptation of air traffic to unexpected changes in the airspace environment, etc. Typical nature and high practical importance of the airspace deconfliction task is the major motivation behind the scope of the research on this Addendum.

Addendum is focused on *investigation and detailed development of a multi-agent architecture* for airspace deconfliction within a homeland security scenario. Existing experience proved that multi-agent framework provides well grounded methodologies, architectures and technologies specifically designated for design and implementation of large scale distributed systems operating in unpredictable environment. That is why this framework is potentially capable to well cope with the task in question.

The Addendum 3 objective is development of mathematical model of the airspace deconfliction system in question and detailed design its multi-agent architecture projected onto software implementation of the latter in the future. The research also assumes analysis of admissible simplifications of the task statement. Development and justification of these simplifications is also a subject of this research.

Three main tasks compose the Addendum 3 scope:

1. Analysis of admissible simplifications and development of the airspace deconfliction task statement.

2. Conceptual analysis of the task, its decomposition and development of the mathematical model of the task. This assumes development of a typical scenario(s) of the airspace deconfliction; decomposition of the task, and development the model of its distributed solving.

3. Development of detailed multi-agent architecture of airspace deconfliction system, its formal specification, as well as detailed formal specification of the target system components.

Although development of any software was not assumed by the Work Plan of the Addendum 3, it was decided to additionally develop small software prototype that would demonstrate feasibility of the basic algorithmic ideas of airspace deconfliction and verify their correctness and efficiency.

The research results that cover all the tasks scheduled for Addendum 3 form the content of this Report.

Principal Investigator of the Task#2 of the Project 1993P
Head of Intelligent System Laboratory
of the St. Petersburg Institute for Informatics and Automation

Prof. Gorodetsky, Vladimir

Ph.D., Doctor of Tech. Sciences

Report Summary

The Project objective is development of mathematical model of the airspace deconfliction system in question and detailed design its multi-agent architecture projected onto software implementation of the latter in the future. The research also assumes analysis of admissible simplifications of the task statement. Development and justification of these simplifications is also a subject of this research. Thus, the main tasks of the Addendum 3 are as follows:

1. Analysis of admissible simplifications and development of the airspace deconfliction task statement.
2. Conceptual analysis of the task as a whole, its decomposition and development of the mathematical model of the task. This assumes development of a typical scenario(s) of the airspace deconfliction, decomposition of the task, and development of a model of its distributed solving.
3. Development of detailed multi-agent architecture of airspace deconfliction system, its formal specification, as well as detailed formal specification of the target system components.

Respectively, *Chapter 1* considers conceptual aspects of the airspace deconfliction task within homeland security scenario. It analyses typical structuring of the airport airspace determining all admissible trajectories of landing and take–off of airliners, formulate rules determining use of this airspace according to existing regulations of air traffic control in the vicinity of an airport. An example of the developed abstract airport airspace structure is further used as a case study for verification and validation of the basic ideas of the airspace deconfliction. The second important issue considered in this Chapter is justification of basic assumptions and problem statement simplifications that are reasonable at the current stage of the research on the problem in question. The assumptions and simplifications used in this research are needed to understand the essence and nature of main challenges peculiar to airspace deconfliction problem. The third issue of conceptual level highlighted in this Chapter is description of the used deconfliction task typical scenario that intends for development of grounded decomposition of this task, discovery of the main roles of the entities participating in solving of the airspace deconfliction task and their functionalities as well as character of the above entities interaction in distributed problem solving. This scenario is further used as a basis for the whole task decomposition and its formal specification in terms of multi-agent architecture. In general, this Chapter forms conceptual basis for the main solutions developed in regard to the design and implementation issues of multi-agent airspace deconfliction system.

Chapter 2 introduces the main algorithmic and design solutions concerning the project of a multi-agent airspace deconfliction system at meta–level. It proposes formal specification of the airport airspace structure in terms of so-called Scenario Knowledge Base framework developed by the research authors and justify the main advantages of its use within the application in question. Actual state of Scenario knowledge base represents jointly formal specification of the airport airspace structure and operation of airliners (both "normal" and hijacked ones). Within such representation of the constraints imposed by the structuring of airspace and rules of air traffic control this formal specification is done in terms of Scenario knowledge base whereas dynamics of admissible airliners' movement is determined as a result of specifically developed inference mechanism. Using the above formal specifications of the airport airspace structure and movement of airliners, an algorithm of airspace deconfliction is described. This algorithm was developed specifically for distributed solution of the deconfliction task while assuming that it is implemented within multi-agent framework. While using conceptual solutions outlined in Chapter 1 and formal structure of the airport airspace representation and reasoning within it, a meta-model of multi-agent airspace deconfliction system is designed. It specifies formally meta-model formalizing the roles of entities participating in airspace deconfliction, protocols of their interaction with emphasis on type of messages which they exchange with, and also assignment of the discovered roles to agents classes. This design and designed formal specification of the meta–model of the multi-agent airspace deconfliction system is done according to the worldwide known Gaia methodology supported by MASDK 3.0 software tool, Multi-agent System Development Kit, implementing the above methodology in graphical style.

Chapter 3 continues in depth development of the design project of multi-agent airspace deconfliction system. It starts from the formally specified meta–model of the system in question (see Chapter 2) and then design in depth the services to be provided by each agent class performing the assigned roles. These specifications determine agent classes behavior in terms of the list of services provided by each agent, their functionalities and algorithms implementing services (like it is assumed by Gaia methodology). Formal specification of services is done in terms of state machines. Materials described in Chapter 2 and Chapter 3 present complete implementation–oriented design project of a multi-agent airspace deconfliction system that is the main objective of the research on Addendum 3. Design of the multi-agent system in question is carried out with support of graphical means of MASDK 3.0 software tool.

Chapter 4 outlines simple software prototype designed for verification and validation of the airspace deconfliction algorithm and graphical representation of the results in real–time mode.

Conclusion summarize shortly the main Addendum results

Chapter 1. Airspace Deconfliction Domain and Problem Analysis

Abstract. This Chapter considers conceptual aspects of the airspace deconfliction task within homeland security scenario. It analyses typical structuring of the airport airspace determining all admissible trajectories of landing and take–off of airliners, formulate rules determining use of this airspace according to existing regulations of air traffic control in the vicinity of an airport. An example of the abstract airport airspace structure developed is further used as a case study for verification and validation of the basic ideas of the airspace deconfliction. The second important issue considered in this Chapter is justification of basic assumptions and problem statement simplifications that are reasonable at the current stage of the research on the problem in question. The assumptions and simplifications used in this research are needed to understand the essence and nature of main challenges peculiar to airspace deconfliction problem. The third issue of conceptual level highlighted in this Chapter is description of the used deconfliction task typical scenario that intends for development of grounded decomposition of this task, discovery of the main roles of the entities participating in solving of the airspace deconfliction and their functionalities as well as character of the above entities interaction in distributed problem solving. This scenario is further used as a basis for the whole task decomposition and its formal specification in terms of multi-agent architecture. In general, this Chapter forms conceptual basis for the main solutions developed in regard to the design and implementation of the multi-agent airspace deconfliction system.

1.1. Topology of Airport Vicinity Airspace

A topology of an airport area space described below corresponds to an abstract airport, which, on the one hand, is intended for demonstration of the basic notions, used in the Project, components of the airspace and their structuring, and, on the other hand, this topology is used below in the case study developed for demonstration of the airspace deconfliction algorithms implemented within a software prototype. This demonstrational topology was developed based on the study of the topologies of the

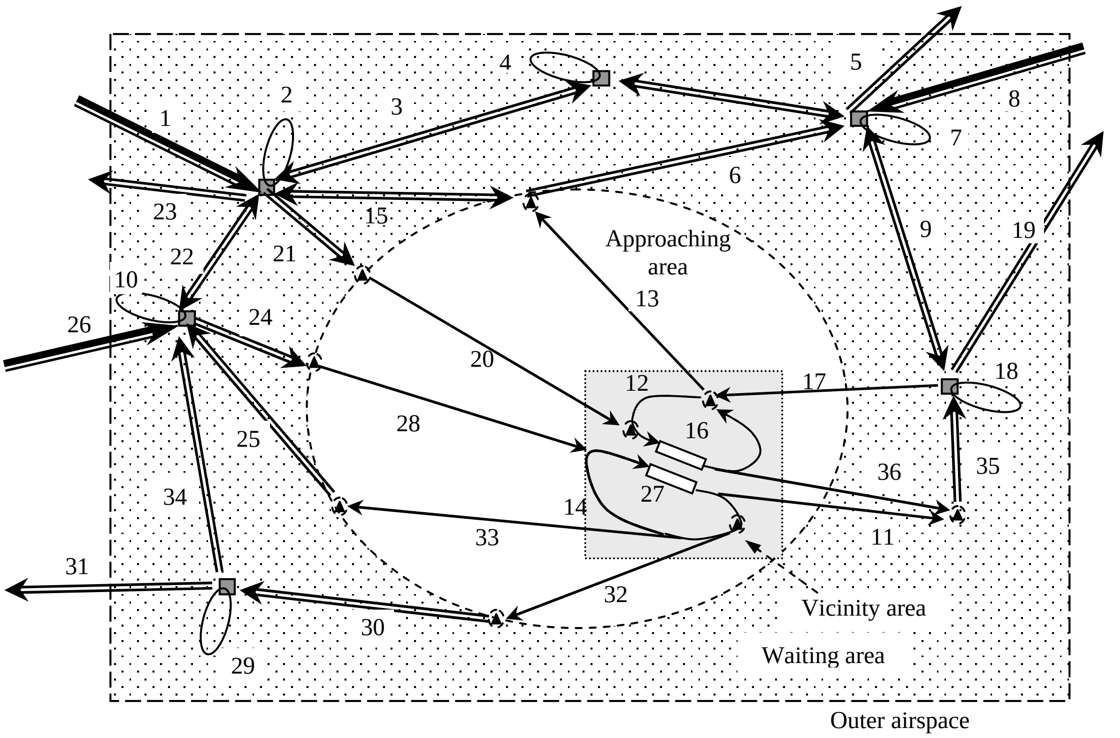
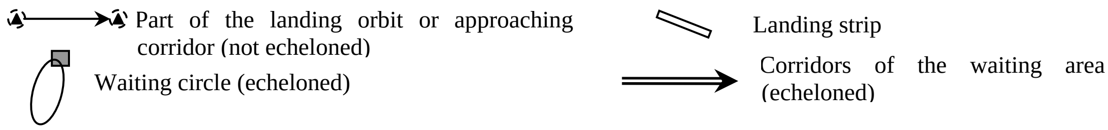


Fig.1.1. An example illustrating the airspace structure (topology) in surroundings of an Airport



Moscow airports [Moscow], airport "Pulkovo" [Pulkovo][1] in St. Petersburg (Russia) and also based on consultation provided by experts from St. Petersburg University of Civilian Aviation (Russia). Materials of the Report [Report 97] were also carefully studied and used in development of a typical airport airspace structure.

Within the surroundings of an airport (within *airport area*, for short) three types of areas are discerned:

1. Vicinity of the airport;
2. Approaching area and
3. Airport waiting area

Airport vicinity area comprises (a) one or several *landing strips*, (b) one or several *landing orbits* for each landing strip. Landing orbit is understood as part of trajectory from approaching area leading directly or through a circle to landing strip. Landing and taking off may be organized in two opposite directions, but during a fixed period of the airport operation both of them, landing and taking off, are mandatory carried out only along one of the admissible directions. Correspondingly, during the aforementioned period, all *normal* landing and taking off airliners have to move along landing strips and landing orbits in the same direction.

Approaching area comprises several routs (*corridors*) connecting waiting area with landing orbits. Each of these paths begins in a fixed *exit point* of the waiting area and ends in the fixed *entry point* of a landing orbit. A corridor is a part of 3D space specified as a rectangular "pipe" within the 3D space having fixed height, length and width. The form of corridor "pipe" is determined by of the form of "pipe" central line having given 3D coordinates of its begin and end.

Airport waiting area comprises several waiting zones (*corridors, waiting circles*). Each of these zones may be entered through one or more *entry points* admitted by the waiting area topology. Airport waiting area zone can be entered either from the *outer airspace*, or from the approaching area. Each corridor of the waiting area is determined in 3D space in the same way as corridor of approaching area. Maximal and minimal heights of each corridor of the waiting area are constant along the central line (i.e. these corridors are assumed horizontal). The corridors along which the normal airliners can move within waiting area are predefined (this is not the case for hijacked airliner). Some of the zones of the waiting area can be assigned orbits along which airliners may move during waiting. While moving from one zone of waiting area to another one, an airliner can change the echelon with the neighbor one, either upper or lower.

An abstract example illustrating the aforementioned notions describing airspace topology in an airport area is given in Fig.1.1.

1.2. Modeling Assumptions and Simplifications

Modeling Assumptions

1. Movement of airliners in the outer space is not considered.
2. The airliners arrive from outer airspace to the airspace of the airport area through given entry points (from fixed sectors of directions).
3. Airport has two landing strips (runways) with one landing orbit per landing strip.
4. An airliner entered into landing orbit has finally either to land or leave landing orbit through a permitted corridor of the approaching area.
5. An airliner entered into an approaching path has to enter in the landing orbit or in the landing strip.
6. An airliner entered into landing orbit has either to land or leave it using a corridor of approaching area.

[1] Both mentioned web sites are presented in Russian.

7. An airliner entered into a corridor of the waiting area may either wait along a waiting orbit of a fixed echelon, or transit into neighbor echelon of the upper or lower corridor, or transit into other corridor, or transit in permitted entry point of the approaching area.

8. Only one airliner may be situated in each particular corridor, waiting orbit, landing orbit and on landing strip (runway).

9. Approaching corridors or landing orbits are not echeloned, i.e. can be occupied by only one airliner.

10. If a corridor or an orbit is occupied by an airliner, it is prohibited for transition of another airliner. It is assumed that topology of the airport area is organized in such a way that the last condition guarantees that airliner minimal separation standards are met, thus providing safety.

11. Fuel content of each airliner including hijacked one is given in terms of time it can fly safely.

12. Hijacked airliner (HA) can follow along any trajectory.

13. Distances to the closest neighbor airports are known and presented in terms of time needed to approach them. These airports are operating in normal mode and can be used at any time.

Modeling Simplifications

1. All airliners including hijacked one have the same constant velocity and rate of climb capabilities.

2. Each corridor, waiting and landing orbit in the airport area airspace is assigned a time interval an airliner spends while moving from its entry point to exit one. This simplification is implied by the previous one.

3. The hijacked airliner can move along any path and at any height and does not agree its movement with the air traffic dispatcher, but in airspace deconfliction it is assumed that in the future the hijacked airliner will be moving rectilinearly and uniformly.

4. Each corridor and orbit are determined by coordinates of entry and exit points of its central line.

5. The prohibited area around the hijacked airliner which normal airliners have to leave as soon as possible is determined as follows: the space within the rectangular tube of given horizontal and vertical sizes (e.g. 10 miles and 500 m. respectively) and situated ahead of hijacked airliner up to fixed length (for example, the length of 10 miles).

6. Transition of a normal airliner from its current position (state, node) to other ones is forbidden if the target position:
    - is occupied currently by other airliner;
    - is overlapping with the prohibited area;
    - is forbidden by the airport area topology; (see, for example, Fig.1).

7. Planning, scheduling, plan performance, etc. are *event-driven*.

8. As events, the following facts are considered in the airspace deconfliction model:

    a. Receiving of information about hijacked airliner by normal airliners (from dispatcher);

    b. Receiving of information by normal airliners (from dispatcher) about changing of hijacked airliner course.

    c. Transition of a normal airliner in new corridor, new waiting or landing orbit, etc. like this (to reflect this information in representation of current situation;

    d. Appearance of a new normal airliner in the airport area;

    e. Disappearance of a normal airliner from the airport area (due to landing or transition in the outer space).

    g. disappearance of the hijacked airliner (independently of the reason)

9. Time interval needed for agents providing pilots with airspace deconfliction assistance is negligibly small in comparison with the time between events implying re-planning of airspace deconfliction.

10. Hijacked airliner is not aware of positions and courses of other airliners operating within airport area.

1.3. Typical Scenario of Airspace Deconfliction

Airspace deconfliction (deconfliction planning, scheduling and performance) starts when an event indicating the fact that an airliner within airport area is hijacked. This event arrives to normal airliners from the air traffic dispatcher. The task is completed if (a) all the normal airliners are landed or left the airport area airspace; or (2) hijacked airliner disappeared in a way.

Airspace deconfliction task is solved according to the following scenario.

1. *Dispatcher* is (first of all but not only) responsible for notification of all the airliners operating within airport airspace area about appearance of a hijacked airliner. This notification is sent to all "normal" airliners operating in the airport airspace. Along with the fact of hijacking, dispatcher sends to all "normal" airliners information about (1) positions of all "normal" airliners and currently being executed plans of "normal" airliner represented in terms of sequences of corridors, waiting and landing circles, etc. assigned the times of entering into and exit out of each aforementioned components of airliner's route; (2) current position and velocity course of the hijacked airliner and also sends (3) information about zones of airport topology to be occupied by hijacked airliners during computed time intervals. These zones are further considered as forbidden for use by normal airliners during the corresponding time intervals. These zones and time intervals are computed by dispatcher according to information about the trajectory of hijacked airliner and existing separation standards for such case. Communication of air traffic dispatcher and pilots of "normal" airliners is performed according to a protocol called below *NHA–protocol*, *Notification about Hijacked Airliner* protocol.

When the notification about hijacking and associated information has arrived to "normal" airliners, the deconfliction task solution is initiated. It is solved with minimal intervention of the air traffic dispatcher. This task is solved via collaboration of pilot using corresponding software. Pilots solve this task via negotiations with each other and with some external entity according to several protocols conceptually outlined below.

2. While having information about (1) airport airspace topology (see Fig.1.1), (2) position and course of the hijacked airliner and (3) current positions of each normal airliner, as well as (4) information about aforementioned forbidden zones mapped time intervals, each pilot (in fact, an assisting software agent)computes several his own best admissible routs resulting in achievement of the airliner goal and selects one of them (as a rule, the most preferable route selected according to a criterion). When selection of the preferable route is done, each pilot requests to reserve "*resources*" (sequence of corridors, waiting and landing circles, etc.) needed to realize the selected route. Along with each requested resource composing the route sequence, particular pilot indicates the real time interval when each resource composing the route is needed to be reserved. The requests of all pilots are sent to the special software program called *Resource Manager* which is considered as *resource owner.* Although it is not located on any "normal" airliner, this software is not a mean of dispatcher as well. Here it is not important where and at which computer this agent is located. At this stage, interaction of pilots and Resource management agent is performed according to *JSV_protocol – Joint Schedule Verification protocol*.
3. When the resource requests are collected from all pilots, *Resource Manager* determines the set of "conflicts" and the list of the pilots (airliners) involved in each of them. A "conflict" is understood as potential violation of one or several rules of regulation determining the safety policy of airliners behavior within the airport airspace area including the necessity to avoid going through the forbidden zones during given time intervals. *Resource Manager* analyzes the conflicts and, according to set of rules, selects one of them that has to be resolved first of all and informs the corresponding pilots about this solution according to *Conflict Resolution Ordering protocol*.
4. The set of assistants of the pilots whose airliners are conflicting for a resource within given time interval negotiate to resolve potential conflicts according to *Conflict Resolution protocol*, *CR-protocol*. This protocol is most important and at the same time it is most complicate.

Sub-scenarios 3 and 4 are repeated to resolve all the conflicts.

5. The schedules of airliners' coordinated behaviors resulting from the above procedures (resulting from repeatable performance of sub-scenarios 3–4) are then forwarded to the responsible dispatcher for verification and approval. Schedule verification is realized by dispatcher through fast simulation. The results of verification (approval or rejection indicating "weak places" of the developed coordinated schedule) are then forwarded to the pilots. This task and associated agent interactions are performed according to *DC_protocol - Deconfliction Completion protocol*.
6. After joint schedule approval (if any) is received, the normal airliners are carrying out the developed (and approved) plan according to the developed (and approved) schedule autonomously, while informing dispatcher and each other (according to an event-driven agent interaction protocol) about the events associated with the behavior of the normal airliners. These events indicate the facts of entering into and exit out of a corridor, echelon, waiting or landing circle thus supporting the global situational awareness of all airliner pilots (assistants of the pilots), dispatcher an *Resource manager* trough their software assistants. This information exchange is supported by *Situational Awareness protocol*, *SA-protocol*.
7. Plan and schedule of the deconfliction task are re-designed if (a) an event informing about new course (and position) of the hijacked airliner has arrived. When such information is received by dispatcher, he cancels the previously resource reservations according to *CRR_protocol - Clear Resource Reservations protocol* and the airspace deconfliction task is solved again; and (b) if the airliner plans are not approved by dispatcher..

1.4. Airliner Behavior Specification

Each "normal" airliner has to move in the airport airspace according to a plan that guarantees avoidance of "conflicts" with other airliners (including hijacked one) through meeting separation standards at any time. Plan can be represented as a sequence of (echeloned) corridors, waiting circles, landing circles, etc. mapped time interval lasting from the time of entrance to and till the time of exit out of the corresponding component of the airport airspace topology. In this Addendum, each aforementioned component of the airspace is considered as a resource, which, during any time point, may be used only by a single airliner. An example given below in Fig.1.2 represents a plan of an airliner behavior that is moving from the corridor 1 to the corridor 16. (The latter corresponds to the landing strip, see Fig.1.1.)

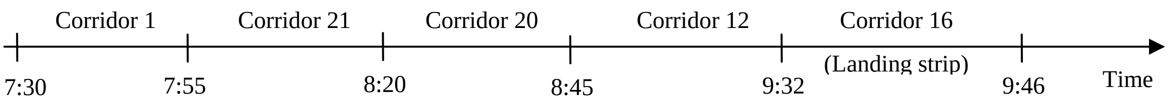


Fig.1.2. Formal specification of an airliner behavior (trajectories along the airport airspace structure)

Thus, while knowing coordinates of each resource, it is possible to compute the airliner coordinates in airport airspace at any time point to the error of its deviation from the central corridor line. Let us recall that the coordinates of a corridor central line are known as well as the coordinates of entry and exit points of this corridor.

The trajectory of movement of the hijacked airliner is represented by its coordinates, course and velocity at the moment when the fact of hijacking is detected. According to the simplifications (see section 1.2) the rout of the hijacked airliner can be computed by using liner equations as follows:

$$x = V_x * (t - t_0) + x_0$$
$$y = V_y * (t - t_0) + y_0 ,$$
$$z = V_z * (t - t_0) + z_0$$

where V_x, V_y, V_z denote the components of the airliner velocity, x_0, y_0, z_0 – coordinates of the airliner at the beginning time point, t_0–time of the hijacked airliner detection and t –current time. Thus, while knowing the trajectory of the hijacked airliner and coordinates of every resources of the airport airspace, it is easy to compute which resources and during which time intervals will be occupied or intersected by the hijacked airliner. The latter added information about separation standards assumed for this case makes it possible to compute time intervals when the resources are forbidden for use by "normal" airliners.

1.5. Admissible Plans and Conflicts

"Conflict" between two or more airliners moving along with corridors of airport airspace structure is understood as the fact when either two or more airliners occupy the same corridor during the overlapping time intervals or separation standards are broken. A plan of an airliner is admissible if it has no conflict with any airliner including hijacked one.

The set of airliners plans are admissible if they have no conflicts.

1.6. Conceptual Problem Statement

Goal of the airspace deconfliction task within the airport airspace is to provide either (a) safe landing or (b) leaving the airport area by normal airliners provided that all of the leaving airliners have enough fuel to land in an other airport independently of trajectory of hijacked airliner. Criterion basis of the aforementioned task has to provide resolving of an airspace deconfliction problem as fast as possible while taking into account fuel capacity of each normal airliner, meeting the minimal airliner separation standards and safety policy described above and with minimal intervention of the dispatcher into airspace deconfliction task solving. This task should be entrusted to the software assisting the pilots of "normal" airliners themselves.

Airspace deconfliction task (planning, scheduling and schedule execution) is initiated by an event informing the participating entities about appearance of a hijacked airliner within airport airspace (or in its vicinity). This event is issued by air traffic dispatcher responsible for safe landing and take-off of airliners within the airport airspace. The airspace deconfliction task is completed when all aforementioned airliners either (1) have landed, or (2) have left airport airspace, or (3) the hijacked airliner has disappeared according to any reason.

1.7. Conclusion on Chapter 1

The main results of this Chapter are as follows:

1. Development of an abstract but typical structure of an airport airspace that is then used as a case study for justification of the main conceptual and design solutions concerning multi-agent airspace deconfliction system.

2. Development and justification of the main assumptions and simplifications of the airspace deconfliction problem that, nevertheless, preserve the main peculiarities of the task in question.

3. Development of a typical scenario of airspace deconfliction task within homeland security scenario that determines basic entities, their roles and coordination in distributed solving of the task.

4. Development of the conceptual statement of the airspace deconfliction problem.

Chapter 2. Meta-level Design Project of Multi-agent Airspace Deconfliction System

Abstract. This Chapter introduces the main algorithmic and design solutions concerning the project of a multi-agent airspace deconfliction system at meta–level. It proposes formal specification of the airport airspace structure in terms of so-called Scenario Knowledge Base framework developed by the research authors. Actual state of Scenario knowledge Base represents jointly formal specification of the airport airspace structure and operation of airliners (both "normal" and hijacked ones). Within such representation of the constraints imposed by the structuring of airspace and rules of air traffic control specification is done in terms of Scenario Knowledge Base whereas dynamics of admissible airliners' movement is specified in terms of specifically developed inference mechanism. Using the above formal specifications of the airport airspace structure and movement of airliners, an algorithm of airspace deconfliction is described. This algorithm was developed specifically for distributed solution of the deconfliction task, while assuming that it is implemented within multi-agent framework. While using conceptual solutions outlined in Chapter 1 and formal structure of the airport airspace representation and reasoning within it, a meta-model of multi-agent airspace deconfliction system is designed. It specifies formally meta-model that formalizes the roles of entities participating in airspace deconfliction, protocols of their interaction with emphasis on type of messages which they exchange with, and also assignment of the discovered roles to agents classes. This design and designed formal specification of the meta–model of the multi-agent airspace deconfliction system is done according to the worldwide known Gaia methodology supported by MASDK 3.0 software tool, Multi-agent System Development Kit, implementing the above methodology in graphical style.

2.1. Airspace Deconfliction Scenario Specification

Let us first specify formally the structure of the airport airspace, determine constraints imposed by the above structure and decide how, in general, the airliners could plan their own safe movement. Fig.1.1 illustrates how the airport airspace is structured and thus the fist task of the design is formal specification of the above structure. As a formal framework used for formal specification of the airport airspace structure, Scenario knowledge base framework proposed in [FinRep-2005] is used. Let us outline the airport airspace structure in terms of scenario knowledge base framework.

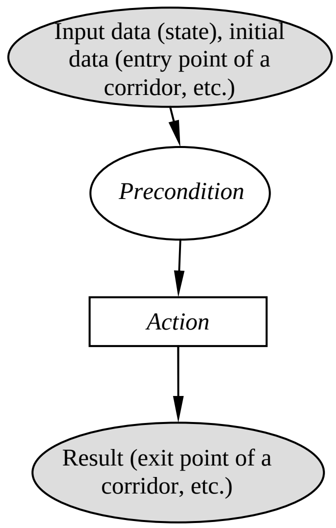


Fig.2.1. Graphical representation of a Simple Behavior Model

2.1.1 Scenario knowledge base

Scenario Knowledge base (SKB) is used in this Addendum for formal specification of the constraints imposed by the rules determining constraints implied by the existing airport airspace structure and thus determining admissible trajectories of the airliners in "normal" situations. It is important to note that this SKB ignores potential conflicts between airliners that may occur if at least a couple of airliners finds out in the same element of the airport airspace topology (structure). Resolving of such conflict is a separate task to be solved according to a special algorithm. This algorithm takes into account initial *airliners configuration*[1] and temporal aspects of its evolution. Thus, structure of the airport airspace is specified by SKB determining admissible trajectories of airliners (have in mind analogy with railway tracks) whereas airliners configuration at given time instant determines positions of all airliners at this time instant (have in mind analogy with positions of the trains on railway track network at a given time instant).

[1] *Airliners configuration* at a time point is understood as the set of positions in terms of airport airspace structure elements (corridors, waiting circle, etc.) of the airliners operating within airport airspace at this time point.

Let us recall [FinRep-2005] that the basic notion of the SKB framework is *Simple Behavior Model* that is constituted by the following elements:

1. *Input state*, or, what is the same in its essence, input data; in the task in question this notion corresponds to an element of the airspace structure (corridor, landing circle, etc.) interpreted as a resource to be used by an airliner;
2. *Precondition*, which determines admissibility of the resource usage; it could be formalized as a set of rules, a set of formulae of the first order logic, etc. Within the airspace deconfliction task aprecondition is true if (a) the corresponding resource is not blocked (is free for use) at the entry time and during the time of its usage; and (2) the entering airliner possesses the needed fuel capacity in a trajectory achieving the final goal if the airliner uses this resource. Let us note that the element "precondition" is intended for taking into account the availability of the "resource" during a time interval what is determined by airliners configuration at given time moment and its evolution during some time interval;
3. *Action* that is movement of an airliner along this element of the airport airspace structure from an entry point to corresponding exit one.
4. *Result*, or, what is the same, o*utput of action*; like "input data" result corresponds to the position of an airliner achieved by it as a result of the above action; for example, a result could be achievement of the entry point of an element of the airport airspace structure.

Depending on application, the listed elements of the Simple Behavior Model may conceptually be interpreted in other ways. In this Report, the aforementioned interpretations are used.

Scenario knowledge Base consists of a structured set of Simple Behavior Models, at that structuring is made via identifying and uniting the same states (entry points of the corridors). In the developed SKB software tool, this procedure is carried out automatically by means of the software tool. An example of SKB corresponding to the airport airspace structure given in Fig.1.1 is presented in Fig.2.2.

One of the basic tasks to be solved in the airspace deconfliction problem that uses SKB formal model is search (inference) for scenarios possessing some specific properties, for example, leading from an entry point of an element of the airport airspace structure to the exit point of other such element. The developed SKB software tool is provided with several variants of inferences, i.e.:

- "*Top–down*" *inference* ("*forward chaining*"); it starts from the known position ("state") of an airliner at the starting time point. (Let us note that hijacked airliner may be situated out of the airport airspace structure.) Additional data have to determine the truth values of the preconditions in time. This specification determines formally the constraints (achievability of that or this corridor and admissibility of their usage depending on whether each corridor is occupied or free of use during given time interval or, at least, during a part of the above time interval). "Top–down" inference is capable to determine all admissible paths (sequences of actions) and sequences of achievable states (entry points of the elements of the airport airspace structure) for given initial configuration of airliners and truth values of preconditions during some time intervals. It is important to note that if an airliner use "top–down" inference to plan its trajectory independently of the plans of other airliners, conflicts may occur. Such conflict should be resolved by a particular algorithm (this algorithm is described conceptually in the next section). Thus, "top–down" inference ("forward chaining") results in computation of all the admissible sequences of the airport airspace structure elements (corridors, waiting circles, etc.) leading from given position ("state") of an airliner through intermediate ones to achievable ones (intermediate positions are also achievable). It is important to note that only some of the achievable positions may correspond to the desired ones forming the set of airliner goals. Therefore, "top–down" inference may result in redundant set of achievable positions.

"Top–down" inference is organized as a step by step procedure. Let us denote the set of positions obtained from the state X_0 at the step k (i.e. by using no more than k inference steps) as $\boldsymbol{X}^{\boldsymbol{Top\text{-}D}}(X_0, k)$.

- "*Bottom–up*" *inference* ("*backward chaining*"); this type of inference starts from a goal (target, or desired position of an airliner within the airport airspace structure) or from a set of such desired positions to be achieved. It is also performed step by step. At every step, it determines a set of positions from which the target position (or a set of desired positions) is achievable, while taking into account the airport airspace structure represented in terms of SKB and truth value of preconditions of actions to be carried out along the path to the desired positions. At the first step, "bottom–up" inference determines the set of the airliner positions from which the desired set is achievable, while using no more than one action. At the k-th step, it determines the set of airliner positions, from which the desired positions are achievable, while using no more than k actions. Let us denote this set as $\boldsymbol{X}^{\boldsymbol{B\text{-}Up}}(X_d, k)$, where X_d is a desired position of airliner. If several positions united into a set $\boldsymbol{X}_d$ are desired then the set $\boldsymbol{X}^{\boldsymbol{B\text{-}Up}}(\boldsymbol{X}_d, k)$ corresponds to the set of positions inferred by k-step "bottom–up" procedure. The latter is interpreted as the set of starting airliner positions from which the desired set $\boldsymbol{X}_d$ of airliner position is achievable, while using no more than k actions.
- *Combined inference* alternating "top–down" and "bottom–up" steps (each of them may be used as first). This inference consists in joint use of "top–down" (starting from the initial airliners position) and "bottom–up" (starting from the set of desired positions) inferences. This inference may lead to the situations when either $\boldsymbol{X}^{\boldsymbol{Top\text{-}D}}(\boldsymbol{X}_0, k) \subset \boldsymbol{X}^{\boldsymbol{B\text{-}Up}}(\boldsymbol{X}_d, k)$, or $\boldsymbol{X}^{\boldsymbol{B\text{-}Up}}(\boldsymbol{X}_d, k) \subset \boldsymbol{X}^{\boldsymbol{Top\text{-}D}}(\boldsymbol{X}_0, k)$. In both situations, the basic steps of combined inference are completed and at final step only the trajectories that pass through the positions which set is determined as $\boldsymbol{X}^{\boldsymbol{Int}}(X_0, \boldsymbol{X}_d, k) =$ $= \boldsymbol{X}^{\boldsymbol{Top\text{-}D}}(X_0, k) \cap \boldsymbol{X}^{\boldsymbol{B\text{-}Up}}(\boldsymbol{X}_d, k)$. Further details of the final step of combined inference are omitted here, because it may be organized in several ways, which need to be validated through simulations.

Thus, use of SKB formalization of the airport structure specification and a variant of inference (combined inference is preferable because it is more efficient) make it possible to generate the sequences of admissible trajectories (sequences of actions, i.e. transitions). These formal models together with the deconfliction algorithms and interaction protocols of distributed entities involved in the airspace deconfliction task (pilots' software, dispatcher's software and resource manager's software) establish a foundation for multi-agent airspace deconfliction system designed below.

2.1.2. Deconfliction Algorithm

Below the developed and partially implemented (for verification and validation purposes) airspace deconfliction algorithm is described in semi-algorithmic manned. Some aspects associated with its multi-agent implementation are omitted here. This very important aspect is described below in the sections devoted to the design of the agent interaction protocols.

Input data:

- A_1, A_2,..., A_N – N "normal" airliners operating within the airport airspace;
- t_0 –value of the time at which the positions of airliners are given (below it also corresponds to the time of a hijacked airliner appearance);
- $X_i(A_j, t_0, \Delta(i, j, t_0))$ –position of j-th airliner within airport airspace structure (as a rule, all of them are situated within elements of the airport airspace structure), where X_i –the name of corridor occupied by the airliner A_j at the time t_0 and $\Delta(i, j, t_0)$ is the time lasted from the entry of airliner A_j into corridor X_i till the time t_0;
- D_j – desired position of the airliner A_j (or $\boldsymbol{D}_j$ if the desired position is a set of them);

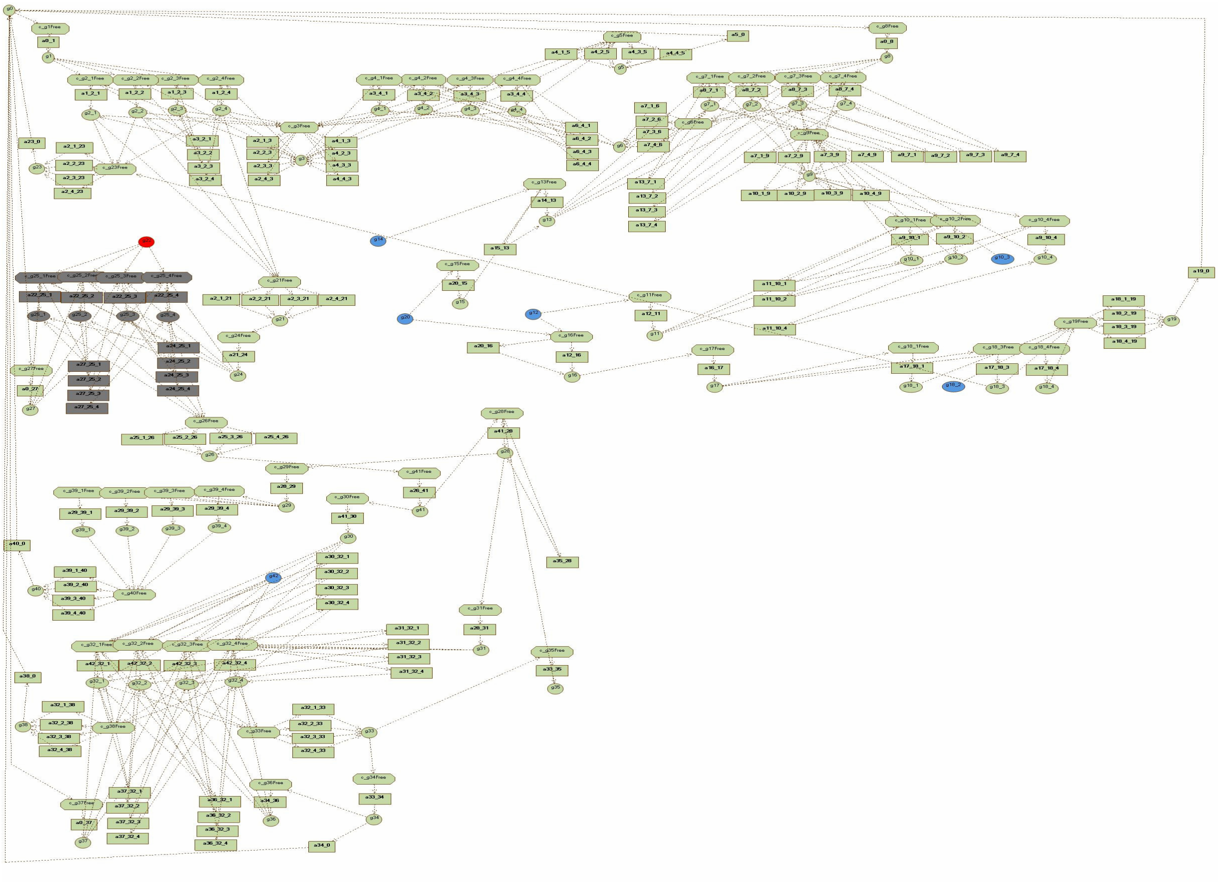


Fig.2.2. Graphical representation of the scenario knowledge base represented as a structure of the set of simple behavior model for the airport airspace structure represented in Fig.1.1.

- P_i –plan of moving of airliner A_j from the current position to the desired position; the set of these plans of all "normal" airliners, $\boldsymbol{P}=\{P_1, P_2, ..., P_N\}$, are conflict–free.
- Formal model of the airport airspace in form of SKB is available for each airliner operating within the airport airspace.
- There is no hijacked airliner;

1. *Appearance of the hijacked airliner*. An event informing about appearance of the hijacked airliner at the time t_0 arrives from dispatcher responsible for air traffic control (hereinafter–dispatcher, for brevity). It is assumed that it is moving in straight-line with a constant velocity and course. This information is broadcasted to all "normal" airliners. While having this information and based of emergency separation standards, dispatcher detects elements of the airport airspace structure which are overlapped by hijacked airliner and computes corresponding time intervals. The detected structure elements are declared by dispatcher as forbidden for use by "normal" airliners during computed time intervals.

2. *Changing of the state of SKB*. While taking into account information about hijacked airliner and forbidden elements of the airport airspace structure, new state of SKB is formed. The SKB change concerns changes of precondition truth values according to the aforementioned information. This new state of SKB may lead to origin of the conflicts in the existing plans of the normal airliners if their routs assume use of forbidden corridors, waiting circles, etc.

3. *Conflict detection*. Existing plans of the airliners are analyzed in order to detect conflicts (if any) with the hijacked airliner. After this, the set of "normal" airliners $\boldsymbol{N}$ is split into two subsets, $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$. The subset $\boldsymbol{N}_1 \subset \boldsymbol{N}$ contains the names of airliners whose plans have no conflicts with the

hijacked one, and the subset $\boldsymbol{N}_2 \subset \boldsymbol{N}$ contains the names of airliners whose plans are conflicting with it.

4. *Search for plans having no conflict with hijacked airliner*. For each $A_i \in \boldsymbol{N}_2$, the set of conflict–free plans (with hijacked airliner) is built. This task is solved on the basis of current state of SKB, in which presence of the hijacked airliner is reflected. Let us note that new plans of the airliners $A_i \in \boldsymbol{N}_2$ may be conflicting with the existing plans of the airliners $A_i \in \boldsymbol{N}_1$. Let us denote the set of *new conflict–free plans* of $A_i \in \boldsymbol{N}_2$ as $\boldsymbol{R}_i$. If the set of plans $\boldsymbol{R}_i$ for airliner $A_i \in \boldsymbol{N}_2$ is empty then a new alternative goal should be selected for it (for example, leaving the airport airspace intended for landing in other airport, or use of the highest echelons), and then, the set of plans $\boldsymbol{R}_i^*$ for alternative goal is computed.

5. *Selection of preferable plan by each airliner*. For each $A_i \in \boldsymbol{N}_2$ the most preferable plan $R_i \in \boldsymbol{R}_i$ or $R_i \in \boldsymbol{R}_i^*$ is selected (for example that plan which realization is optimal in respect to required fuel consumption). Each such plan is represented in terms of a sequence of resources (corridors, circles, etc.) assumed for use and time interval of each resource usage. Further on, airliner A_i sends request to *Resource management* software to reserve needed resources during computed time intervals.[1]. While having reservation requests from all $A_i \in \boldsymbol{N}_2$ received, *Resource manager* checks whether conflicts within the plans of all airliners, both $A_i \in \boldsymbol{N}_1$ and $A_i \in \boldsymbol{N}_2$ exists. Each conflict is mapped the set G_j of airliners involved into conflict number ***j***.

6. *Ordering of conflict resolution*. Resource manager software determines the order according to which the set of detected conflicts should be resolved. This procedure is heuristic–based. The following heuristics are used to order the detected conflicts:

- Time when conflict starts, priority=1: the earlier a conflict occurs the higher is its priority;
- Fuel capacity, priority=2: the less planned fuel reminder at the time of destination of a desired position is the higher of the airliner priority is;
- Total number of airliners involved in the conflict, priority 3: The larger the number of conflicting airliners in a group G_j is the higher is its priority.

7. *"Normal" airliners conflict resolution.* For selected conflict and group G_j of conflicting airliners, the conflict is resolving. Conflict resolution task is solved based of heuristics. In this procedure, the following heuristics are used:

- The rest of the time which an airliner may spend within the waiting circles in its way to the desired position up to the time of the fuel exhaust; priority=1: the less the rest of the fuel capacity the higher priority.
- The rest of the fuel at the time of the desired position achievement; priority=2: The less the fuel rest the higher priority.

Preference is paid to an airliner according to rules determined over the heuristic truth values. For winning airliner the resource during requested time interval is reserved.

Item 7 is repeated while all the conflicts between the airliners of group G_j are resolved. After this ,the next conflict for next group G_{j+1} is resolved in the same way.

[1] Corresponding protocol is described below

2.1.3. Deconfliction Algorithm in Case of Hijacked Airliner Course Change

This algorithm is organized in the same way as algorithm described in previous subsection with the only difference that is caused by new state of SKB since new course of hijacked airliner results in new elements of the airport airspace that have to be forbidden for use by "normal" airliners and normal airliners situated in new positions. New state of SKB is computed by dispatcher software. Before initiation of the algorithm in question, the followings have to be done:

- Cancel all previously imposed reservations of the resources caused by existence of hijacked airliner having previous course;
- Compute new plans of "normal" airliners starting from the current positions with ignorance of new blockings of resources (names of resources and forbidden time intervals of their use) since they are not so far computed. It is necessary because resource availability is changed after change of course of the hijacked airliner.

2.2. Roles of Multi-agent Airspace Deconfliction System

This section presents specification of the airspace deconfliction system components in terms of roles and the tasks performed by them as it is assumed by Gaia methodology implemented within Multi-agent System Development Kit 3.0 used, it turn, for multi-agent system technology support. This specification takes into account multi-agent architecture of the system implementation with emphasis on distributed character of performance.

2.2.1. Role: Pilot Assistant

The main functionalities of this role are (a) generation of the current plan for achievement of the desired position; (2) re-planning if a conflict is detected; (3) negotiation with the dispatcher (more precisely, with the dispatcher assistant role) (4) negotiation with the resource manager role, (5) negotiation with the assistants of the pilots of other "normal" and also (6) providing the assisted pilot with some important information, in particular, information about impossibility of conflict–free achievement of the desired position. For this role, the identifier "*Pilot*" is further used.

2.2.2. Role: Resource Owner

The main functionalities of this role are (1) management of the airport airspace resource usage, (2) detection and prevention of the conflicts and (3) participation in negotiations with pilot assistant and dispatcher assistant roles according to a number of protocols outlined below. For this role, the identifier "*Resource Owner*" is further used.

2.2.3. Role: Dispatcher Assistant

Dispatcher assistant role is responsible for (1) informing all the participants of the airspace deconfliction system (performing *Pilot* and *Resource Owner* roles) concerning detection of the hijacked airliner and attributes of its movement and modification of the hijacked airliner course, (2) coordination of the re-planning procedures performed by *Pilot* role and (3) simulation of the coordinated airliner behavior for verification of its validity and approval of the latter. For this role, the identifier "*Dispatcher*" is further used.

2.3. Meta–model of Multi-agent Airspace Deconfliction System

The aforementioned meta–model comprises declaration of the roles, protocols and agent classes as well as declaration of their interaction. Let us outline this meta–model illustrated in Fig.2.3.

Agent classes (left hand upper part of Fig.2.3)

1. *PA_agent* (*Pilot Assistant agent*) – an agent class assisting the pilots and performing the "*Pilot*" role. The total number of instances of this agent class is equal to the number of "normal" airliners operating in the airport airspace.
2. *RM_agent* (*Resource Management agent*) – the agent class responsible for management of the airport airspace resources. It performs the role of the *Resource_Owner*. In the system the single

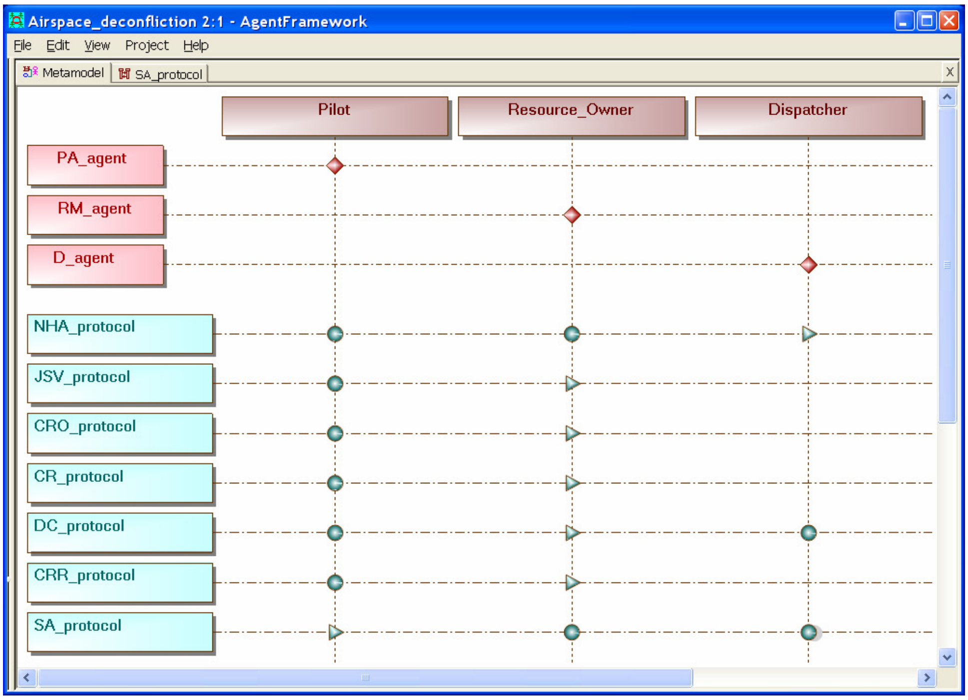


Fig.2.3. Meta–model of multi-agent airspace deconfliction system specified by the means of MASDK 3.0.

instance of this agent class responsible for management of the total resources of the airport airspace is used.

3. *D_agent* (*Dispatcher agent*) –the agent class performing the role *Dispatcher*. In the system single instance of this agent class is used.

Allocation of the roles to the agent classes.

This allocation is presented in Fig.2.3 and it was described above.

Protocols (left hand bottom space of Fig.2.3)

NHA_protocol (*Notification about Hijacked Airline protocol*). It supports notification of the components of the multi-agent airspace deconfliction system about appearance of a hijacked airliner. *Dispatcher* role initiate this protocol performance intended for notification of agents performing the *Pilot* and *Resource_Owner* roles and also for sending to them information about hijacked airliner, its position and course. This protocol also sends to them information about resources to be forbidden for use by "normal" airliners and time intervals when forbidden resources are unavailable.

JSV_protocol (*Joint Schedule Verification protocol)*. *Resource_Owner* initiates this protocol performance. It is intended for verification of the existing plans of the "normal" airliners with regard to the existence of conflicts with the hijacked airliner as well as with the other "normal" airliners. According to their plans, agents performing the *Pilot* role inform agent performing the *Resource_Owner* role about needed resources mapped the time intervals of their usage.

/Note: At the first step, *Resource_Owner* role analyzes only existence of the conflicts betwen plans of normal" airliner and hijacked one, because it is assumed that, till this time, the plans of "normal" airliners are conflict-free./

CRO_protocol (*Conflict Resolution Ordering protocol)*. This protocol is used by *Resource_Owner* role for imposing an order according to which the detected conflicts are to be resolved. This is done in dialogs with agents performing the *Pilot* role. Practically, *Resource_Owner* role selects a conflict to be resolved at the forthcoming step mapped the list of conflicting airliners.

CR_protocol (*Conflict Resolution protocol)*. This protocol is intended for detection of an airliner which should be granted the preference to use the requested resource for which other airliners are also

compete during the same or overlapping time interval. The participants of this protocol are conflicting instances of agent class *PA_agent* of the group G_j.

DC_protocol (*Deconfliction Completion protocol)*. When the conflicts are resolved, instance of the agent class *D_agent* verifies and approves the proposed plans of the "normal" airliners (if any) and sends to the latter permission to execute the proposed plans.

CRR_protocol (Clear Resource Reservations protocol). This protocol serves for canceling of the resource blocking when the hijacked airliner changes its course. This information is sent to the instances of the *PA_agent* agent class.

SA_protocol (Situational Awareness Protocol). The instances of *PA_agent* agent class inform each other and instances of *RM_agent* and *D_agent* classes about their transitions from one element of the airport airspace structure to other ones.

2.4. Formal Specification of the Protocols

2.4.1. *NHA_protocol*

Airspace deconfliction system performance is initiated by this protocol. According to it, *Dispatcher* role informs other participants of the task solution about appearance of hijacked airliner and forward to them corresponding information. Additionally, *Dispatches* sends list of forbidden resources mapped to time intervals when these resources are unavailable. Fig.2.4 represents this protocol in terms of entities involved in its performance and types of messages they exchange with. The latter are as follows:

- *Inform_airliners* message type– it is sent from *Dispatcher* role to all agent instances performing the *Pilot* role. This message is intended to inform airliners about appearance of a hijacked airliner and forbidden resources. While receiving this message, instances of *PA_agent* agent class transform current state of their SKB recording in it information about resources forbidden for use during given time intervals.
- In*form_resource_owner* message type– it is sent from *Dispatcher* role to *Resource_Owner* role to inform the latter about appearance of a hijacked airliner and resources forbidden for use. While receiving this information, *Resource_Owner* records in its mental model information about unavailable resources during given time interval.

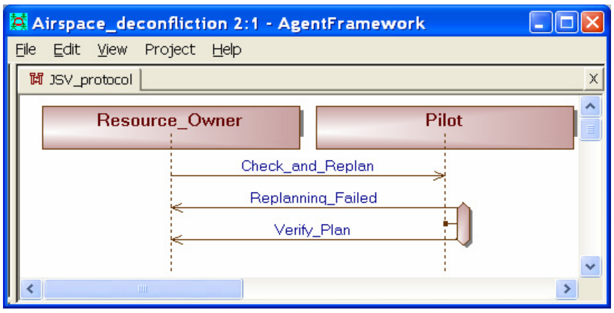


Fig.2.5. *JSV_protocol* specified by the means of MASDK 3.0..

2.4.2. *JSV_protocol*

This protocol is intended to change the plans of airliner, while taking into account the information about unavailability of the resources to be occupied by hijacked airliner and reserve resources for new plans.

Types of messages:

- *Check_and_Replan* message type– *Resource_Owner* sends query to all instances of *PA_agent* agent class, while asking them to check their current plans in order to detect potential conflicts, if any, and re-plan their trajectories if necessary. The latter checks their plans according to current state (after receiving information about hijacked airliner) of own SKB. If it detects conflict(s) it re-plans own movement using inference mechanism of SKB.
- *Replanning_Failed* message type– this type of message is used by instances of *PA_agent* agent class to inform *Resource_Owner* about detection of conflicts and existence of no admissible plans (The reasons of absence may be various: too small capacity of fuel, unavailability of waiting resources, etc.). To resolve such kind of conflict, an intervention of the human–dispatcher is required who is responsible for proposal of non-standard solution like use of non-standard corridors

of the airport airspace structure, selection of landing in other non-typical airport (e.g., military airport), etc.

- V*erify_Plan* message type– instances of *PA_agent* agent class request *Resource_Owner* to reserve resources. *Resource_Owner* stores this information in its mental model and uses it later to verify whether there exist conflicts between (new and old) plans of the "normal" airliners.

2.4.3. *CRO_protocol*

While having information about potential conflicts between plans of various "normal" airliners, *Resource_Owner* is responsible for management of these conflict resolutions. As a rule, several conflicts exist and in this case it is important to decide, which conflict is the most important to resolve

Рис.2.6. *SA_protocol* specified by the means of MASDK 3.0.

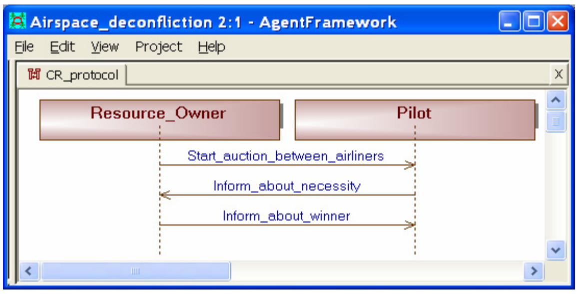


Fig.2.7. *CR_protocol* specified by means of MASDK 3.0.

it first of all. This task is implemented in terms of argumentation–based auction involving bargaining of conflicts taking into account properties of plans of airliners composing each conflict. Criteria of winner (winner corresponds to a conflict selected) were described above.

Types of messages:

- *Inform_About_Conflict* message type– *Resource_Owne*r sends to the instances of *PA_agent* agent class message notifying about existence of conflicts with the plans of other "normal" airliners, while requesting information needed to start auction.
- *Inform_about_situation* message type – instances of *PA_agent* agent class send to *Resource_Owner* the requested information, which then is transformed by the latter into proposals corresponding to each conflict. Then, *Resource_Owne*r simulates auction and determines winner over the set of conflicts, thus determining which conflict has to be resolved first of all.

2.4.4. *CR_protocol*

This protocol supports interaction (in form of argumentation–based auction) of instances of *PA_agent* agent class to select a preferable airliner to whose the resource that is the subject of bargaining should be granted.

Types of messages:

- *Start_auction_between_airliners* type of messages – *Resource_Owner* sends to all instances of *PA_agent* agent class notification about start of the auction intended bargaining for resource inspired conflict.
- *Inform_about_necessity* message type– instances of *PA_agent* agent class send to *Resource_Owner* their arguments justifying importance of

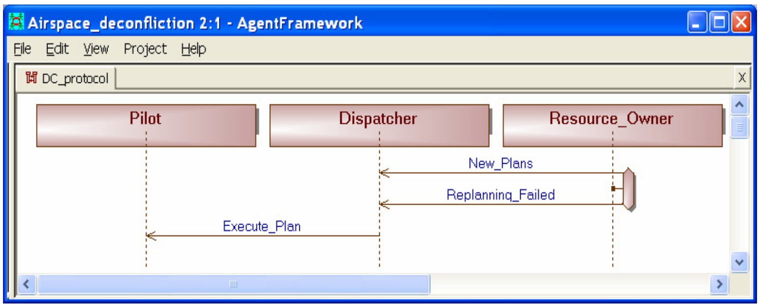


Fig.2.8. *DC_protocol* specified by the means of MASDK 3.0.

conflicting resource for this particular airliner. *Resource_Owner* accumulate these arguments and then, based of heuristic rules, determines winner, i.e. airliner which is granted the resource that was the subject of auction.

- *Inform_about_winner* message type – *Resource_Owner* notifies the instances of *PA_agent* agent class about auction winner. The winner, an instance of *PA_agent* agent class, uses this information to construct its plan of the desired position achievement.

2.4.5. *DC_protocol*

When all the conflicts are resolved, *Resource_Owner* forwards the total plan of moving of all the "normal" airliners operating in the airport airspace to the instance of the *D_agent* agent class for approval.

Message types:

- *New_Plans* message type – *Resource_Owner* sends to the instance of the *D_agent* agent class new plans of "normal" airliners.
- *Replanning_Failed* message type– *Resource_Owner* notifies the instance of the *D_agent* agent class about absence of the conflict–free plans. This information is also forwarded to the corresponding instances of *PA_agent* agent class.
- *Execute_Plan* message type– instance of the *D_agent* agent class notifies the instances of *PA_agent* agent class about approval of their plans and permission to perform them.

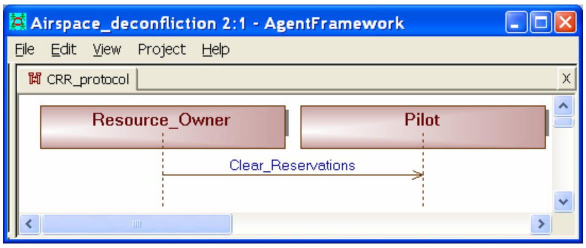


Fig.2.9. CR_protocol specified by the means of MASDK 3.0.

2.4.6. *CRR_protocol*

This protocol is performed in case if hijacked airliner changes its course.

Message type:

- *Clear_Reservations* message type – *Resource_Owner* notifies instance of the *D_agent* agent class about necessity to cancel resource unavailability caused by existence of the hijacked airliner.

2.4.7. *SA_protocol*

This protocol intends to keep informed other components of the multi-agent airspace deconfliction system about current positions of the "normal" airliners when they transit from one element of the airport airspace to other ones.

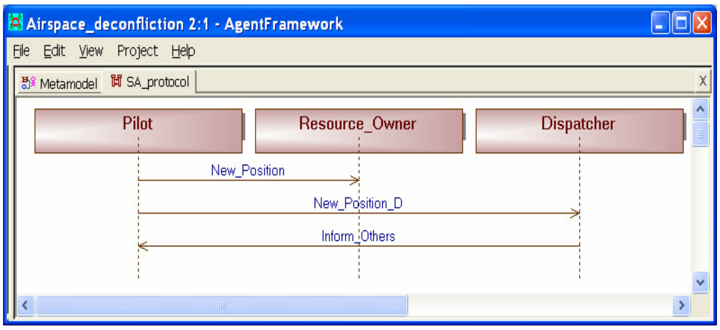


Fig.2.10. CR_protocol specified by the means of MASDK 3.0.

Message types:

- *New_Position* message type – an instance of *PA_agent* agent class notifies *Resource_Owner* about transition into new corridor, waiting circle, etc.
- *New_Position_D* message type– an instance of *PA_agent* agent class notifies the instance of *D_agent* agent class about transition into new corridor, waiting circle, etc.
- *Inform_others* message type– the instance of *D_agent* agent class forwards this information to the instances of *PA_agent* agent class.

2.5. Conclusion on Chapter 2

The results of this Chapter are as follows:

1. Formal representation of the airport airspace structure in terms of Scenario Knowledge Base framework and constrained movement of the airliners in termas of inference within Scenario Knowledge Base framework.

2. Algorithm of airspace deconfliction specifically designed for distributed implementation within multi-agent architecture.

3. Completely designed formal specification of the meta–model of multi-agent airspace deconfliction system representing formally the roles of the system entities solving the airspace deconfliction task in distributed manner, protocol of their ineractions and messages with which they exchange, and basic classes of agents assigned particular roles discovered due to the developed task decomposition.

Chapter 3. Formal Specification of Agent Behavior

Abstract. This Chapter continues in depth development of the design project of multi-agent airspace deconfliction system. It starts from the formally specified meta–model of the system in question (see Chapter 2) and then designs in depth the services provided by each agent class performing the assigned roles. These specifications determine agent classes behavior in terms of the list of services provided by each agent, their functionalities and algorithms implementing each of them (how it is assumed by Gaia methodology). Formal specification of services is done in terms of state machines. Materials described in Chapter 2 and Chapter 3 present complete implementation–oriented design project of a multi-agent airspace deconfliction system that is the main objective of the research on Addendum 3. Design of the multi-agent system in question is carried out with support of graphical means of MASDK 3.0 software tool.

3.1. *PA_agent* agent class

3.1.1. Behavior Meta–Model

Meta–model of any agent behavior determines the set of services provided by corresponding agent and scheme of their use during participation of an agent in performance of a protocol. At the next stage the services are specified formally in terms of particular *state machines*. The following services composing the meta–model of *PA_agent* agent class are provided by it:

- *Clear_Resource_Reservations* – this service is initiated by the message issued by *CRR_Protocol*. It erase records from SKB concerning resource reservation stored in the agent memory.
- *Receive_Hijacked_Airliner_Info* – this service is initiated by the message issued by *NHA_protocol*. It stores data about hijacked airliner and marks the resources it occupies and will occupy.
- *Plan_Verification* – this service is initiated by the message issued by *JSV_protocol*. It checks the existing plan in order to detect conflicts in context of information about hijacked airliner and if a conflict exists the service generates new plan.
- *Conflict_Resolution_Ordering* – this service is initiated by the message issued by *CRO_protocol*. It collects information about conflict represented in input message and conveyed it to the instance of *D_agent* agent class in order to use this information in conflict resolution ordering procedure.

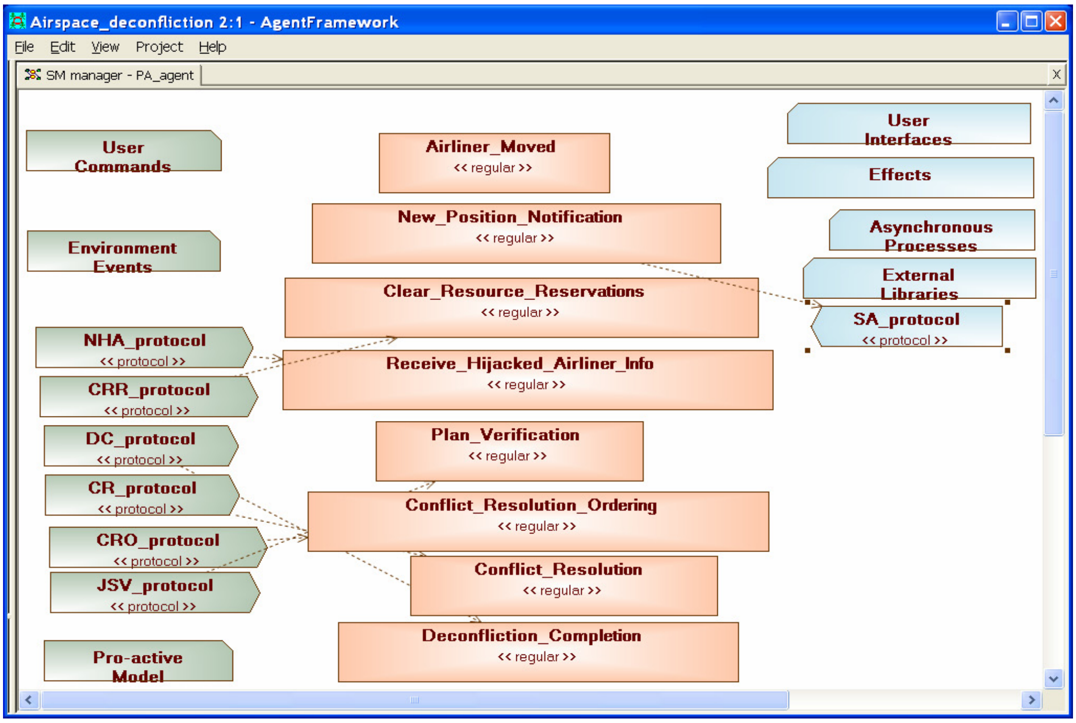


Fig.3.1. *PA_agent:* behavior meta–model specified by the means of MASDK 3.0.

- *Conflict_Resolution* – this service is initiated by the message issued by *CR_protocol*. It collects information about conflict represented in input message and conveys it to the instance of *D_agent* agent class. The latter selects an airliner to be granted the resource that is the subject of a conflict.
- *Deconfliction_Completion* – this service is initiated by the message issued by *DC_protocol*. This message approves new "normal" airliner plan.
- *New_Position_Notification* – this service is initiated by the message issued by *SA–protocol* aiming at notification of the agents about transition of the airliner in new element of the airport airspace structure.
- *Airliner_Moved* – this service receives notification of an instance of the *PA_agent* agent class about its transition into new element of the airport airspace structure.

3.1.2. *PA_agent* Agent Class Service Specification

Clear_Resource_Reservations

This service is specified by the state machine containing single state:

- *Clear* – this state deletes all records about resource reservation made earlier from SKB.

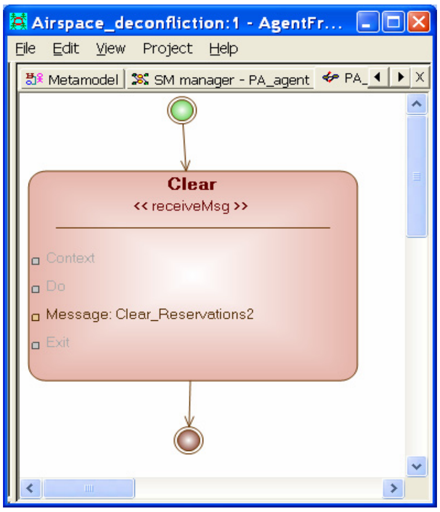


Fig.3.2. Specification of *Clear_Resource_Reservations*. service

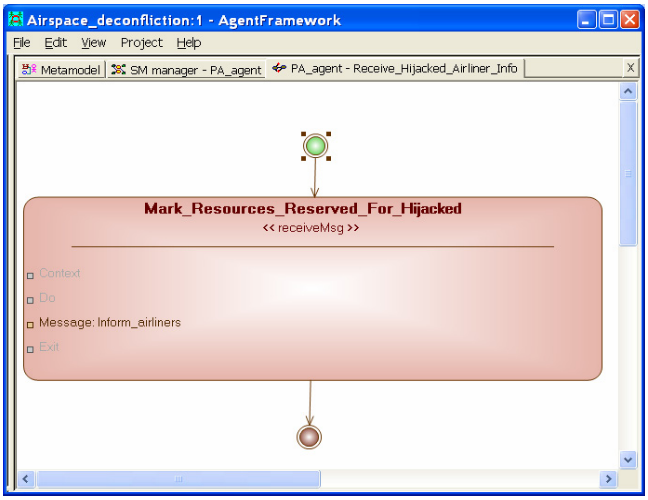


Fig.3.3. Specification of *Receive_Hijacked_Airliner_Info* service.

Receive_Hijacked_Airliner_Info

This service is specified by the state machine containing single state:

- *Mark_Resources_Reserved_For_Hijacked* – this state computes the trajectory of particular airliner, detects availability of the resources and create the records in SKB representing resources captured and to be captured by hijacked airliner.

Plan_Verification

This service is specified by the state machine containing three states:

- *Check_Plan_and_Replan* – this state checks the current plan in

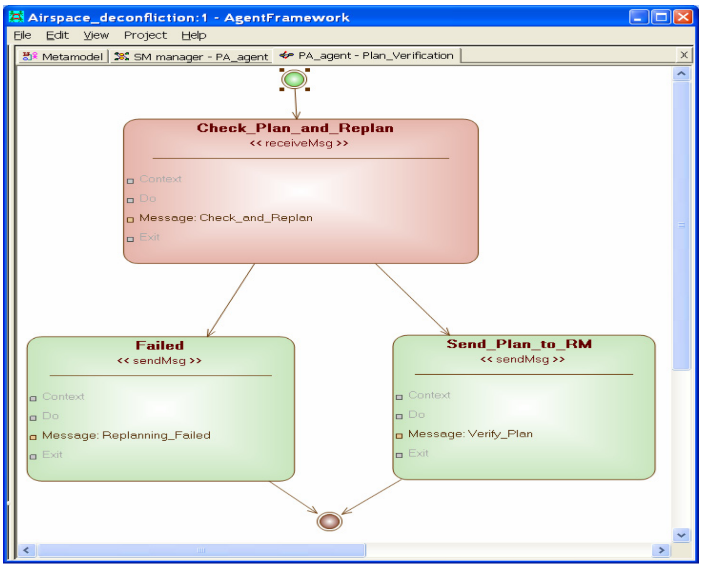


Fig.3.4. Specification of *Plan_Verification* service.

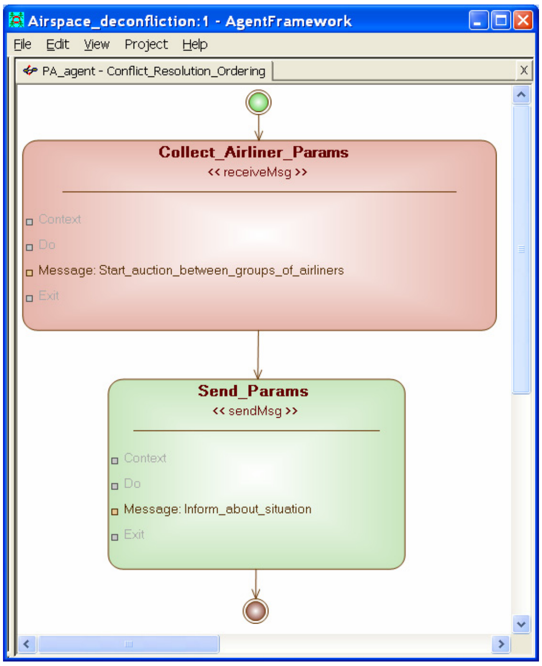


Fig.3.5. Specification of *Conflict_Resolution_Ordering*. Service.

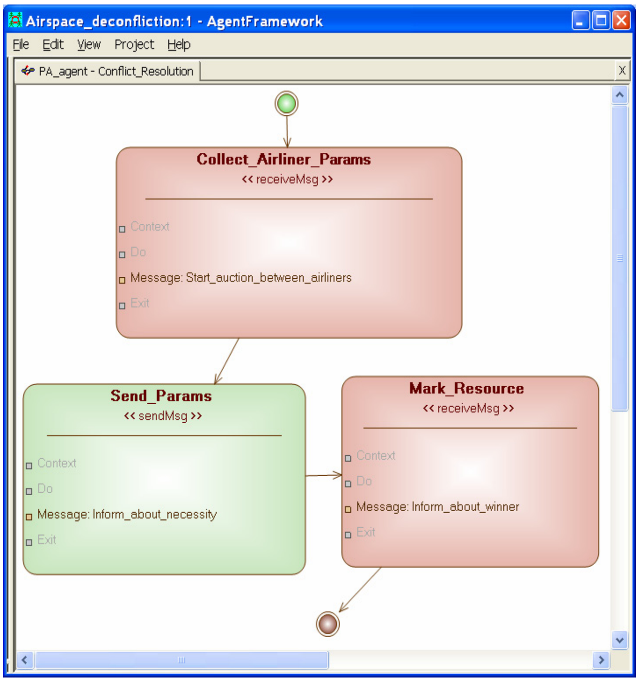


Fig.3.6. Specification of *Conflict_Resolution* service.

order to detect conflicts within SKB. If a conflict is detected this state builds new plan that is not used the reserved resources. This is done by use of inference mechanism dealing with the knowledge represented in current state of SKB.

- *Failed* –this state is executed if conflict(s) is detected but new admissible plan cannot be built. This state is responsible for sending a message notifying about the aforementioned fact.
- *Send_Plan_to_RM* – this state sends message about successfully built new conflict–free plan. Along with notification, new or previously built plan is also sent.

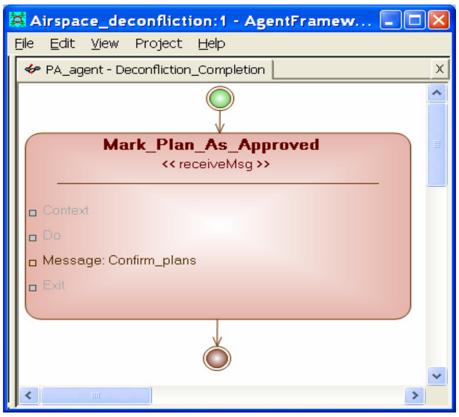


Fig.3.7. Specification of *Deconfliction_Completion* service.

Conflict_Resolution_Ordering

This service is specified by the state machine containing two states:

- *Collect_Airliner_Params* – this state performs collection of the information about the rest of the fuel of the airliner after achievement of the desired position;
- *Send_Params* – this state sends the aforementioned information.

Conflict_Resolution

This service is specified by the state machine containing three states:

- *Collect_Airliner_Params* – this state is responsible for collection of the following information: the rest of the fuel of the airliner after its achievement of the desired position and the time interval that the airliner may fly in the waiting circles before the conflict occurs;
- *Send_Params* – this state is responsible for sending of the collected information;
- *Mark_Resource* – this state receives message containing information about an airliner that win in bargaining for resource. In the airliner SKB (mental model), information about resource reserved is recorded.

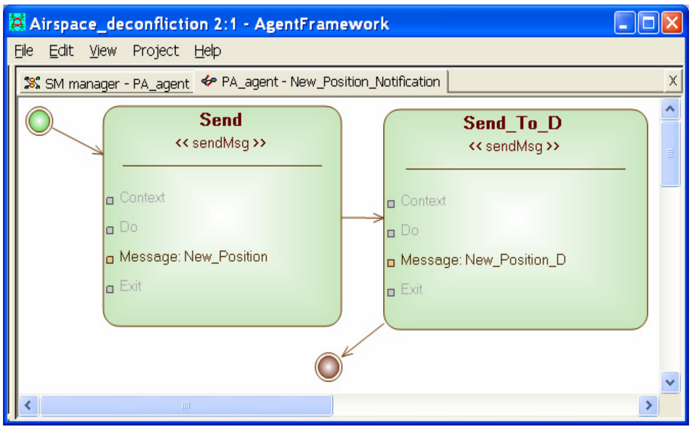


Fig.3.8. Specification of *New Position Notification* service.

Fig.3.9. Specification of. *Airliner_Moved* service.

Deconfliction_Completion

This service is specified by the state machine containing single state:

- *Mark_Plan_As_Approved* – it receives a message approving the proposed plan. After this, the airliner is permitted to perform its plan.

New_Position_Notification

This service is specified by the state machine containing two states:

Send – this state provides notification of the instance of the *RM_agent* agent class about new position of airliner;

Send_To_D –this state provides notification of the instance of the *D_agent* agent class about new position of airliner.

Airliner_Moved

This service is specified by the state machine containing single state:

- *Receive_Info* – this state receives information about transition of an airliner from one corridor, waiting circle, etc. to another one.

3.2. *RM_agent* agent class

3.2.1. Behavior Meta–Model

The following services composing the meta–model of *RM_agent* agent class are provided by it:

- *Process_Hijacked* – this service is initiated by the message issued by *NHA_protocol*. In *RM_agent* mental model, it marks resources occupied and to be occupied by hijacked airliner and initiates conflict resolution procedures.
- *Unblock_Resources* – this service deletes from agent mental model all records corresponding to resource reservation and prepares information to be sent to the instances of the *PA_agent* agent class requiring to cancel the existing resource reservations.
- *Verify_All_Plans* – this service sends queries to all instances of the *PA_agent* agent class to check the existing plans in order to detect conflicts if any.
- *Process_Conflict* – this service processes a selected conflict for winner selection (as a result of a step of argumentation–based auction).
- *Select_Conflict_To_Process* – this service processes all the conflicts in order to select one of them to be resolved first of all (as a result of a step of argumentation–based auction).

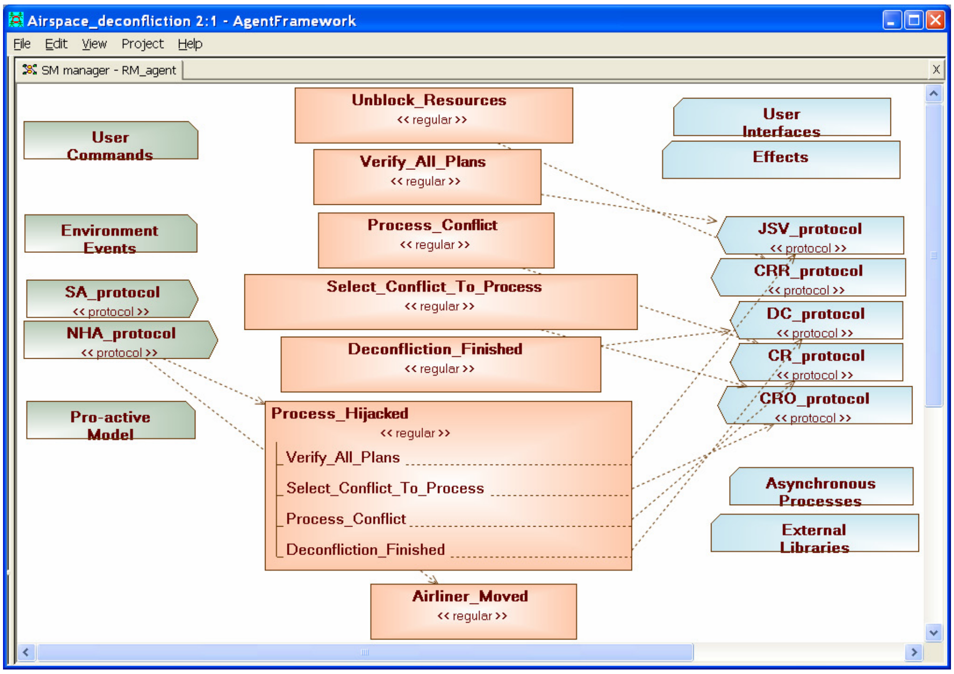


Fig.3.10. *RM_agent*: behavior meta–model

- *Deconfliction_Finished* – this service processes completion of the deconfliction procedure.
- *Airliner_Moved* – this service receives information about transition of an airliner from one element of the airport airspace (corridor, circle, etc.) to another one.

3.2.2. *RM_agent* Agent Class Service Specification

Process_Hijacked

This service is specified by the state machine containing six states:

- *Get_Hijacked_Info* – this state receives from *DA_agent* information about attributes (position, course) of the hijacked airliner.
- *Verify* – this state generate nested query of the *Verify_All_Plans* service.
- *Emergency* – this state is initiated

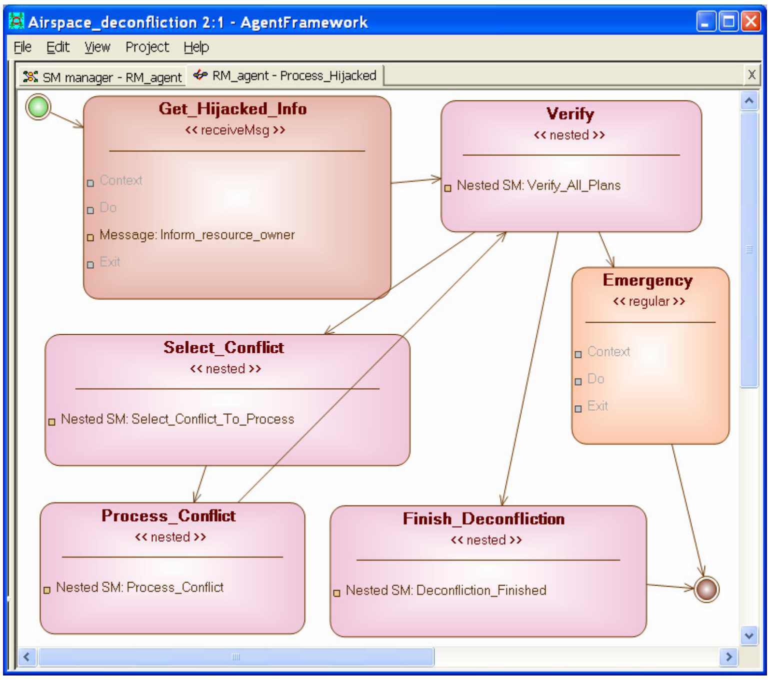


Fig.3.11. Specification of the *Process_Hijacked* service.

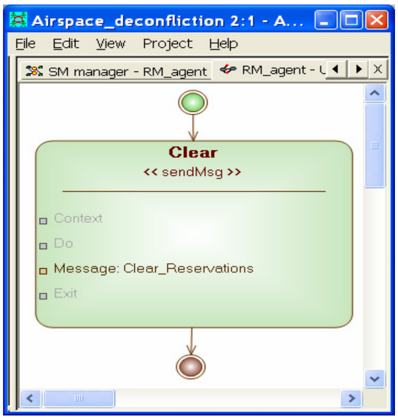


Fig.3.12. Specification of the *Unblock_Resources* service.

if some airliners do not capable to find new conflict–free plans, that means that a conflict requiring to produce a non-standard decision (dispatcher is responsible for such decision). This query is sent to dispatcher and completes the process.

- *Select_Conflict* – this state generates nested query of the *Select_Conflict_To_Process* service.
- *Process_Conflict* – this state generates nested query of the *Process_Conflict* service.
- *Finish_ Deconfliction* – this state generates nested query of the *Deconfliction_Finished* service.

Unblock_Resources

This service is specified by the state machine containing single state:

- *Clear* – this state informs the instances of the *PA_agent* agent class about the necessity to cancel in their mental models information concerning resource reservation.

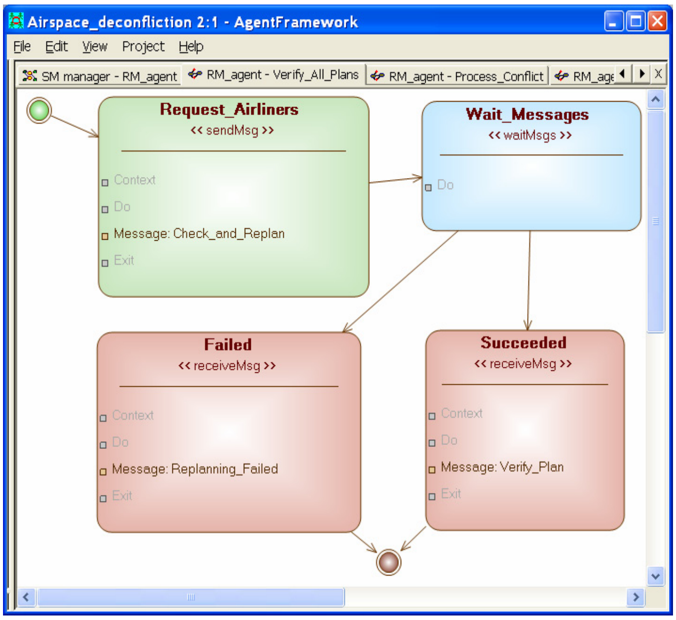


Fig.3.13. Specification of the *Verify_All_Plans* service.

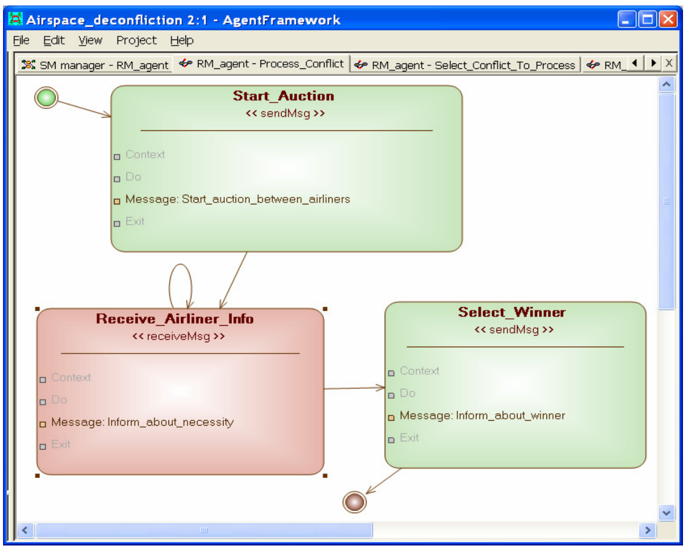


Fig.3.14. Specification of the *Verify_All_Plans* service .

Verify_All_Plans

This service is specified by the state machine containing six states:

- *Request_Airliners* – this state is responsible for broadcasting of the message informing instances of *PA_agent* agent class about the necessity to check and re-plan their trajectories.
- *Wait_Message* – this state implements waiting of the replies from the instances of the *PA_agent* agent class.
- *Failed* – this state is initiated in case if a message about absence of a conflict-free plan is received from an instance of the *PA_agent* agent class. It is responsible for saving of this information in the *DA-agent* mental model.
- *Succeeded* – this state is initiated in case if a message about successful building of the conflict–free plan from an instance of the *PA_agent* agent class is received. The service is finished when such messages have received from all the instances of the *PA_agent* agent class.

Process_Conflict

This service is specified by the state machine containing three states:

- *Start_Auction* – this state initiates auction among instances of the *PA_agent* agent class of group G_j involved in conflict number j. This auction is aimed to decide to which airliner the resource that is the

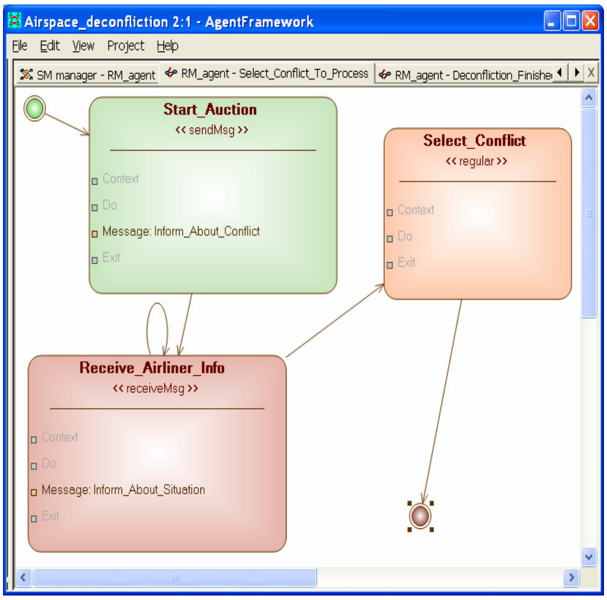


Fig.3.15. Specification of the *Select_Order_of_Conflict_To_Process* service

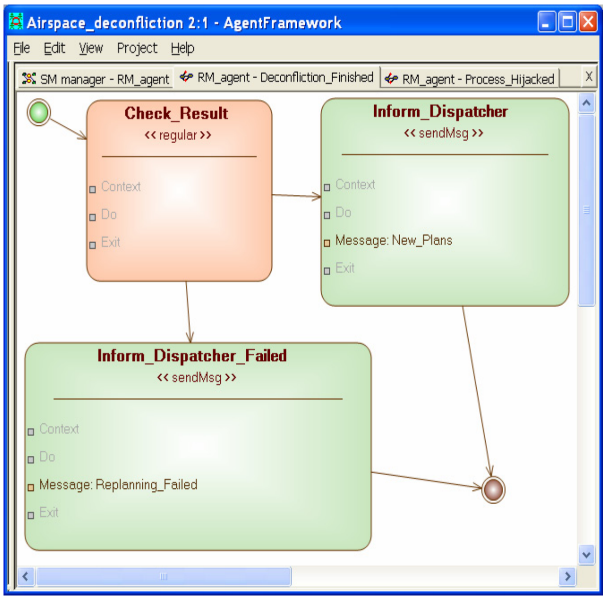


Fig.3.16. Specification of the *Select_Conflict_To_Process* service

subject of conflict is granted.

- *Receive_Airliner_Situation* – this state is responsible for receiving and saving information from particular instance the *PA_agent* agent class.
- *Select_Winner* – this state is responsible for winner selection among the instances the *PA_agent* agent class of group G_j involved in conflict number j and informing all the auction participants about winner, which is granted resource that was a subject of bargaining.

Select_ Order_of_Conflict_To_Process

This service is specified by the state machine containing three states:

- *Start_Auction* – this state initiates the auction among all the instances of the *PA_agent* agent class aiming to select a conflict that should be deconflicted first of all.
- *Receive_Airliner_Info* – this state is responsible for receiving information about conflicts from the instances of the *PA_agent* agent class.
- *Select_Conflict* – this state is responsible to make decision what conflict should be resolved first of all.

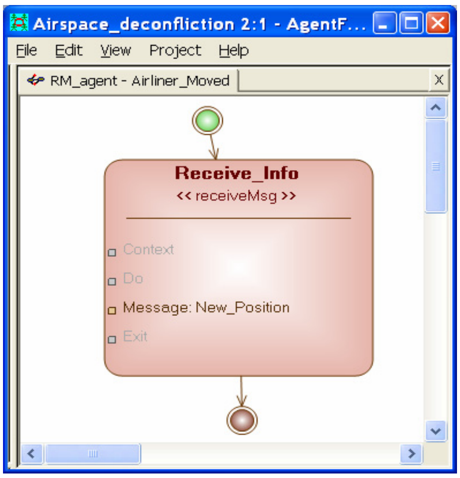


Fig.3.17. Specification of the *Airliner_Moved* service.

Select_Conflict_To_Process

This service is specified by the state machine containing three states:

- *Check_Result* – in this state, *RM_agent* checks the re–planning results.
- *Inform_Dispatcher_Failed* – this state implements the sending information about the absence of a conflict–free plan for a conflict to *DA_agent*.
- *Inform_Dispatcher* – this state is responsible for sending new conflict–free plans of all the airliners to *DA_agent*.

Airliner_Moved

This service is specified by the state machine containing single state:

- *Receive_Info* – this state is responsible for receiving information about transition of an airliner from one element of the airport airspace to another one.

3.3. *D_agent* agent class

3.3.1. Behavior Meta–Model

The following services composing the meta–model of *D_agent* agent class are provided by it:

- *Airspace_Deconfliction* – this service is initiated by dispatcher. He inputs attributes of movement of the hijacked airliner. Then, a procedure analyzing existence of potential conflicts between hijacked airliners and "normal" ones is performed and if a conflict exist the re-planning procedure is initiated.
- *Process_New_Plans* – this service performs processing (simulation) and approval of new "normal" airliner plans.
- A*irliner_Moved* – this service receives information about transition of an airliner from one element

of the airport airspace to another one.

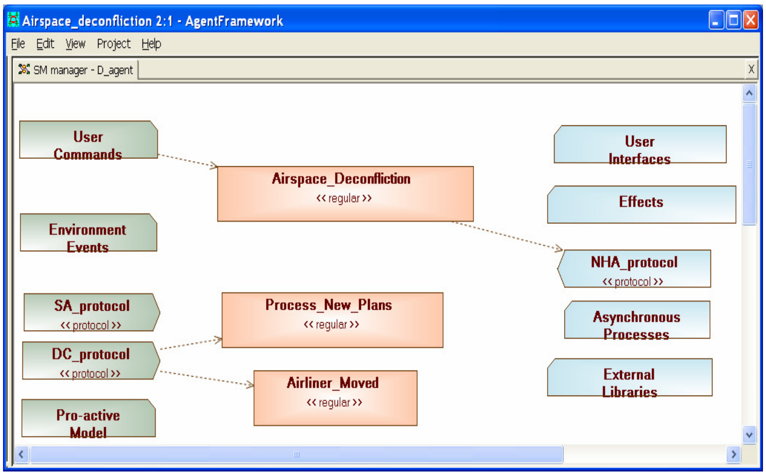


Fig.3.18. *D_agent*: behavior meta-model

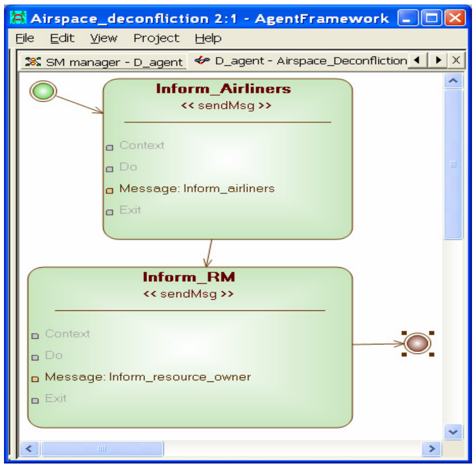


Fig.3.19. Specification of the *Airspace_Deconfliction* service

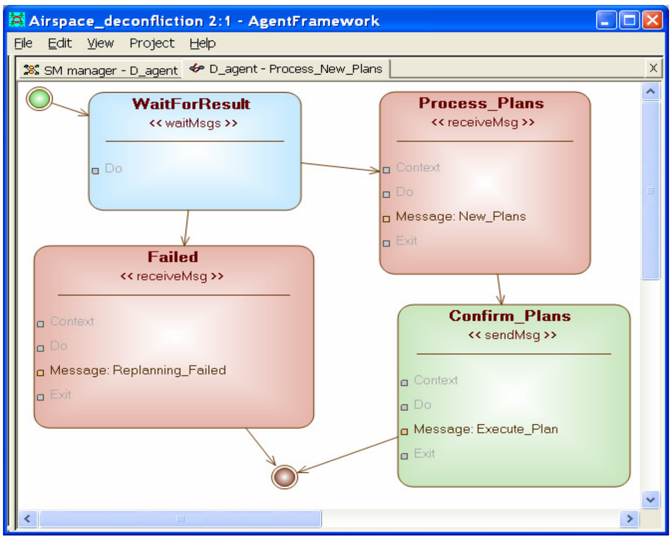


Fig.3.20. Specification of the *Process_New_Plans* service.

3.3.2. *DA_agent* Agent Class Service Specification

Airspace_Deconfliction

This service is specified by the state machine containing two states:

- *Inform_Airliners* – this state is responsible for notification of the instances of the *PA_agent* agent class.
- *Inform_RM* – this state is responsible for notification of the *RM_agent* agent class.

Process_New_Plans

This service is specified by the state machine containing four states:

- *WaitForResult* – this state corresponds to waiting of the *RM_agent* reply.
- *Process_Plans* –this state is initiated if new airliner plans are received. It provides these plans processing.
- *Failed* – this state is responsible for processing of exception when conflict–free plans do not exist.
- *Confirm_Plans* – this state performs broadcasting of the plan approvals to the instances of the *PA_agent* agent class and permission to start their performance.

Airliner_Moved

This service is specified by the state machine containing two states:

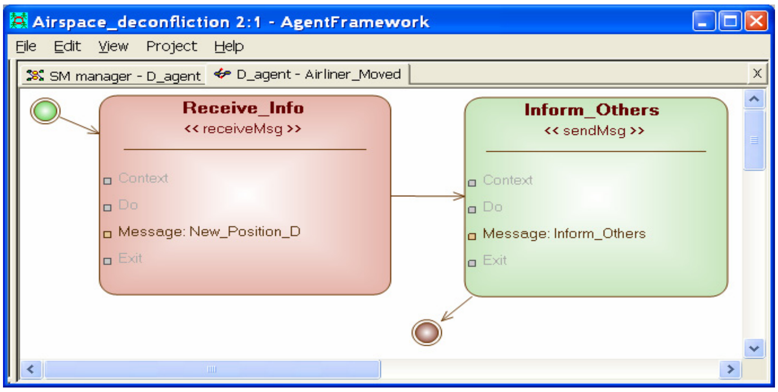


Fig.3.21. Specification of the *Airliner_Moved* service

- *Receive_Info* – this state receives information about transition of an airliner from one element of the airport airspace to another one.
- *Inform_Others* – this state is responsible for re-translation of the received information to other instances of the *PA_agent* agent class.

3.4. Conclusion on Chapter 3

The main result presented in this Chapter, together with the results of previous Chapter, presents complete design project of the multi-agent airspace deconfliction system specified formally in terms of Gaia methodology and developed through use of graphical means of MASDFK 3.0 software tool. This formal specification is represented in XML language, which is generated automatically from its graphical representation developed within MASDK 3.0 software tool.

Chapter 4. Simulation Results: Validation of Deconfliction Algorithm

Abstract. This Chapter outlines simple software designed for verification and validation of the airspace deconfliction algorithm and graphical representation of the results in real–time mode.

4.1. Software Prototype: Simplified Implementation

Software prototyping was intended verification and validation of the basic algorithm of the airspace deconfliction that is planning of collective conflict–free behavior of the "normal" airliners when hijacked airliner appears within airport airspace. This algorithm was described in section 2.1. "*Verification*", which was the main objective of the simulation prototype, is understood here as checking of correctness and convergence of this algorithm, because it is of iterative character. "*Validation*" is understood as assessment of the planning and re-planning algorithm efficiency depending on some parameters like the number of "normal" airliners operating within airport airspace and "complexity" and redundancy of the airport airspace structure. It is important to note that, in practical situations, this algorithm has to be completed during several seconds. An objective was also to develop recommendations in regard how this algorithm should be improved to meet the practical requirements.

The developed software was not organized as multi–agent system (software entities interaction protocols were not implemented). Nevertheless, the basic structure of the algorithm was preserved: instead of agents, the instances of the corresponding C++ software classes and, instead of message exchange, calls of the object methods were used. One of the basic components of the developed software prototype are Scenario Knowledge Base and inference of admissible sequences of airliner transition aimed to achieve the desired position. These components were implemented by use of previously developed (during previous two quarters of the research on the 1993P project) software tool, which makes it much easier implementation of the software prototype as a whole. Let us note that as inference mechanism, the "top–down" (forward chaining reasoning) approach was used.

Separation standards were modeled by a sphere of given diameter with the airliner in the central point. Different separation standards were used for "normal" airliners and hijacked one. In the simulation procedure diameter of the sphere around the hijacked airliner was assigned value of 5 km. However, this value is an attribute of the software and its value can be changed by user of the software prototype.

In the developed software prototype, the following specifications of its components and particular initial data were used:

- Specification of Scenario knowledge base presented by XML file (it is assumed by the developed SKB software tool). This file corresponds to the graphical specification airport airspace structure depicted in Fig.1.1.
- Specification of "normal" airliners initial positions (at the moment when hijacked airliner appears in the airport vicinity) and plans of their moving (landing and take-off) according their goals predefined by air traffic dispatcher.
- Characteristics of trajectory of hijacked airliner at the time of its appearance (position and course).

The results of the simulation concern new conflict–free trajectories of "normal" airliners specified in terms of sequences of transition from one element of the airport airspace structure to another one assigned pair of times corresponding entry to and exit from corresponding element respectively. If conflict–free plan does not exist (was not found by the algorithm) then corresponding message is generated. It is worth to note that practically, the developed algorithm makes it possible to plan "normal" airliners trajectories in the routine situation when a hijacked airliner is absent.

Several case studied differing in the total number of the "normal" airliners operating within airport airspace were simulated. First of all, the algorithm was simulated for small number of such airliners, and later on, as the algorithm was tested and corrected for improving, the number of airliners was increased.

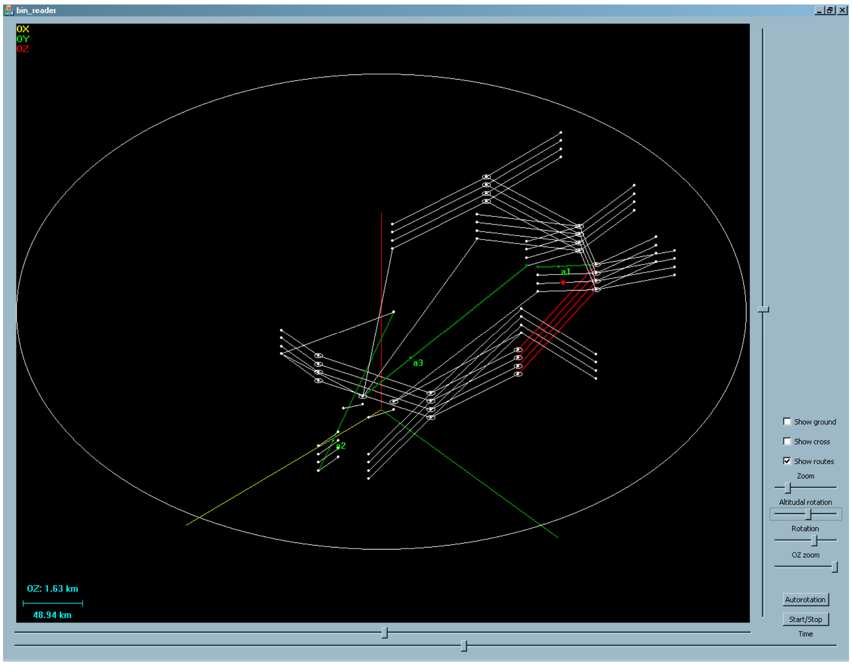


Fig.4.1. Visualization of the situation with 3 "normal" airliners within airport airspace structure (3D projection)

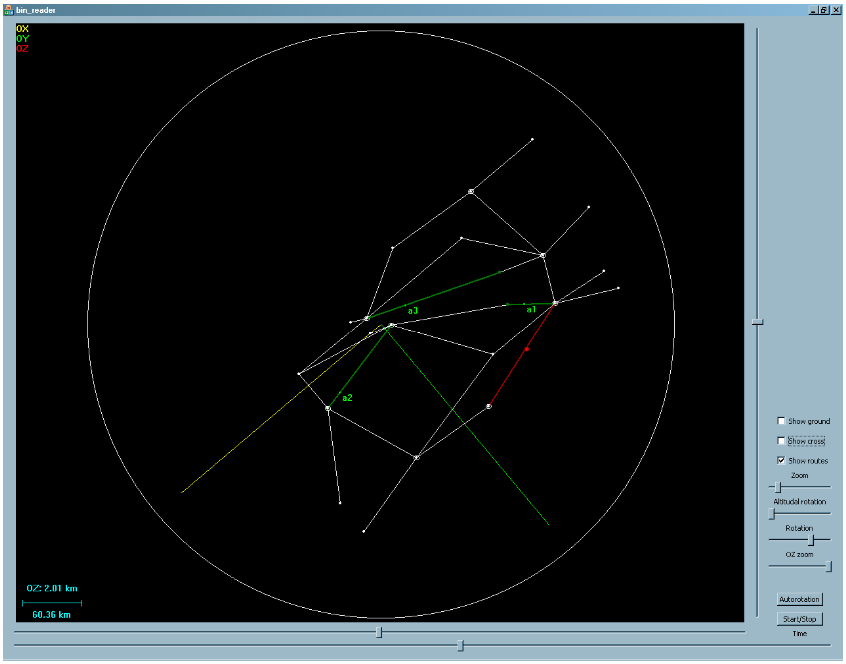


Fig.4.2. Visualization of the situation with 3 "normal" airliners within airport airspace structure (view from above)

Examples of visual representation of a situation at time points are given in Fig.1–Fig.4. Fig.4.1 presents a snapshot of the visualization of a situation within airport airspace in 3D way for the case when three "normal" airliners and single hijacked one are operating within airport airspace structure[1]. Fig.4.2 and 4.3 present the same situation but for vertical view (view from above) and horizontal side

[1] We intend to demonstrate this software prototype when this demonstration is appropriate for US Air Force representatives.

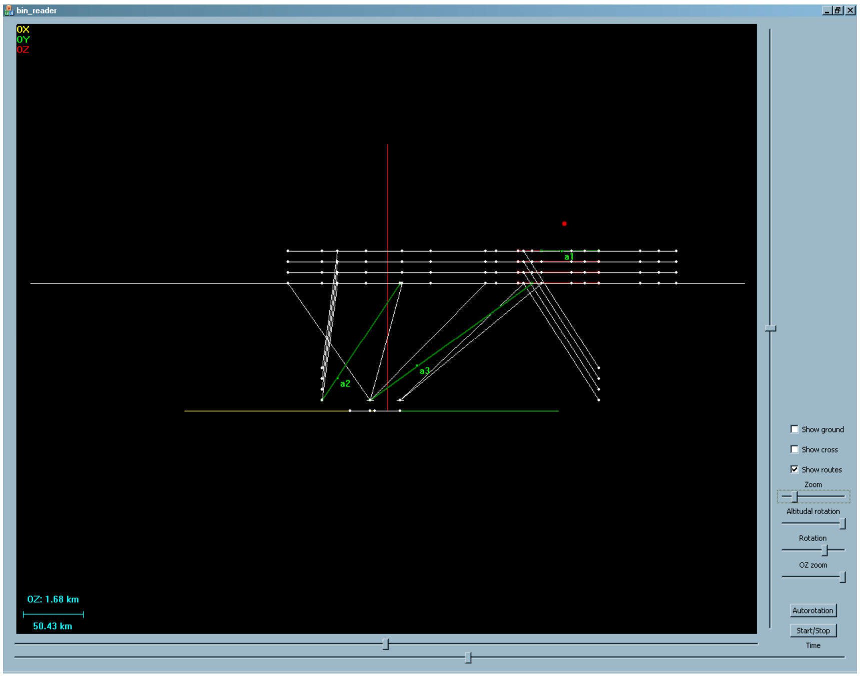


Fig.4.3. Visualization of the situation with 3 "normal" airliners within airport airspace structure (horizontal side view)

views respectively. Fig.4.4 presents a snapshot of the visualization of a situation within airport airspace in 3D way for the case when five "normal" airliners and single hijacked are operation in the

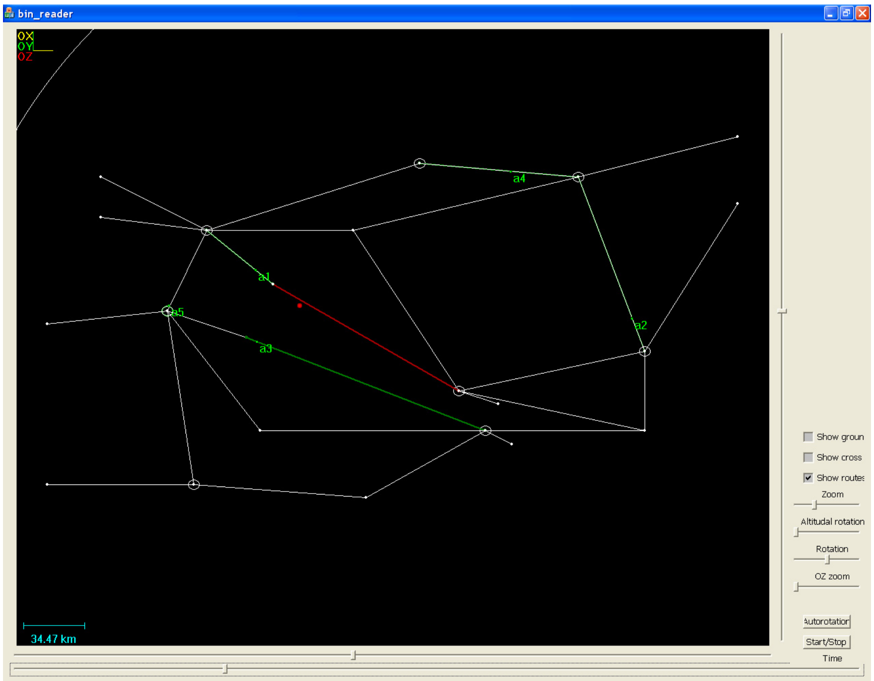


Fig.4.4. Visualization of the situation with 5 "normal" airliners within airport airspace structure (3D projection)

airport zone.

The testing results proved that algorithm is working stable and there was no case when it gets caught in an endless loop. The number of iteration depends on the number of "normal" airliner and experimentally evaluated complexity was close to quadratic. But this cannot so far be generalized on

general case because experiments were too limited. However, it seems that such a complexity may be achieved through use of "good" heuristics. Discovery of such heuristics, evaluation of their impact on speed of algorithm and quality of the resulting plans should be the subject of further research. In this research, machine learning-based approach to heuristics generation may found out very productive.

Of course, the time spent for plans generation has to depend on "complexity" and "dimension" of the airport airspace structure. However, it is not clear so far, how this dependency should look like.

For the airport airspace structure depicted in Fig.1.1[1] and for five "normal" airliners, when a standard Pentium 4 computer is used, the total re-planning time is close to 1 minute. Of course, this time is not so far satisfactory, but there exist a number of ways to speed this algorithm.[2] It is also necessary to take into account the fact that in practice the deconfliction task is solved in parallel by many computers situated in the airliners and in dispatcher working place. This may considerably speed the deconfliction task solving.

4.2. Software Prototype Interface

Developed software prototype was intended for real–time demonstration of the airspace deconfliction system operation. It makes possible visually observe airport airspace and "normal" and hijacked airliners flights in 3D space and quickly simulate and evaluate deconfliction algorithm performance.

The user is provided with the capability to change projection of 3D picture and its scaling up and down. "Normal" airliners are depicted with green circles whereas the hijacked one is depicted with red one. An element of the airport airspace structure (corridor, waiting and landing circles) is colored in green if it is occupied by "normal" airliner whereas if it is occupied by hijacked airliner it is colored in red. If a situation when separation standards between "normal" and hijacked airliners are broken occurs then the corresponding corridor is colored in crimson.

[1] Let us note that this structure should be assessed as very complex. At least, it is more complex than such structure used in particular airports of Moscow and St. Petersburg, Russia.

[2] Let u remind that development of re-planning algorithm or even more so an optimal one is out of the main objective of this project.

5. Report Conclusion

The main result of the Addendum 3 is completely designed multi–agent airspace deconfliction system supporting distributed mode of solving deconfliction task with minimal intervention of the air traffic dispatcher. The design project of this system is formally specified in terms of XML language. Exactly this task was the main objective of the Addendum 3 and it is completely fulfilled.

Particular results of the research on Addendum 3 are as follows:

1. An abstract but typical structure of an airport airspace that was then used as a case study for justification of the main conceptual and design solutions concerning multi-agent airspace deconfliction system was developed. This structure is specified formally in terms of Scenario Knowledge Base framework providing a number of useful properties of the resulting formalization. The most important properties are automatic satisfaction of constraints imposed by airport airspace structure which are resulted from rules and regulations adopted in practice of air traffic control. This structure and its formal specification may be further used as a testbed for investigation, verification and validation of various airspace deconfliction algorithms.

2. The main assumptions and admissible simplifications of the airspace deconfliction problem are analyzed. The proposed problem statement that admits the proposed assumptions and simplifications, nevertheless, preserves the basic features and peculiarities of the real–life situations. On the other hand, the proposed problem statement makes it easier the study of the problem peculiarities and investigation of the potential approaches to as well as algorithm and architectures of airspace deconfliction system design.

3. Developed typical scenario of airspace deconfliction task within homeland security scenario determining basic entities involved in distributed task solving, their roles and coordination of their distributed performance forms grounded conceptual basis for the task decomposition and its design and implementation within multi-agent framework.

4. Joint formal representation of the airport airspace structure in terms of Scenario Knowledge Base and constrained movement of the airliners in terms of inference within Scenario Knowledge Base framework provide potential efficiency of the airspace deconfliction system from real–life perspective.

5. Distributed algorithm of airspace deconfliction specifically designed for multi-agent implementation may be considered as a first version that can be significantly improved due to involving more heuristics, which can be generated based of machine learning approach.

6. Completely designed formal specification of the meta–model of multi-agent airspace deconfliction system representing formally the roles of the system distributed entities, protocol of their interactions and messages, with which they exchange, are typical for various statements of the airspace deconfliction problem and thus can be reused.

7. Formal specification of services provided by agent classes proposed in the designed project of multi-agent airspace deconfliction system represented in terms of state machines also may be reused in the class of applications comprising more general air traffic control problem.

In design of the deconfliction system, an experience accumulated during fulfillment of the Project 1993P was widely used. In particular, experience in development of multi-agent systems for data and information fusion for situation assessment, in use of Gaia methodology for multi-agent system design and implementation as well as experience in multi-agent technology embodied in MASDK 3.0 software tool provide the necessary knowledge, skill and software making it possible to successfully complete this research during two quarters.

Further research should first of all concerns the following topics:

1. Software implementation of the designed multi-agent airspace deconfliction system.

2. Improvement of the distributed deconfliction algorithm via use of more grounded heuristics that may be generated experimentally on the basis of machine learning techniques. Other way of the deconfliction algorithm speed up concerns improvement of the inference mechanism used within SKB for inference of admissible conflict–free plans of "normal" airliners. In particular, it would be

interesting to implement inference mechanism combining "Top–Down" and "Bottom–Up" reasoning within Scenario Knowledge Base framework.

3. Development of more realistic model of airport airspace structure and models of movement of airliners including hijacked one.

4. In depth investigation of the task complexity depending on the number of the airliners operating within airport airspace and on the structure of the latter.

References

[FinRep-2005] Final Report on Extensions 1 and 2 of Project 1993P. May 2005.

[Moscow] http://www.radioscanner.ru/avia/maps/bykovo.zip.

[Pulkovo] http://www.radioscanner.ru/avia/maps/pulkovo.zip.

[Report-97] Safety Analysis of Air Traffic Control Upgrades. Final Report. Project Manager Prof. Nancy Leveson, UW, 1997. http://citeseer.ist.psu.edu/cache/papers/cs/17038/ftp:zSzzSzftp.cs.washington.eduzSzpubzSzsafetyzSzfinal.pdf/leveson97safety.pdf.